

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:

Shigeo ORII

Application No.:

Group Art Unit:

Filed: July 16, 2003

Examiner:

For: PARALLEL EFFICIENCY CALCULATION METHOD AND APPARATUS

**SUBMISSION OF CERTIFIED COPY OF PRIOR FOREIGN
APPLICATION IN ACCORDANCE
WITH THE REQUIREMENTS OF 37 C.F.R. § 1.55**

Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

Sir:

In accordance with the provisions of 37 C.F.R. § 1.55, the applicant(s) submit(s)
herewith a certified copy of the following foreign application:

Japanese Patent Application No(s). 2002-212387

Filed: July 22, 2002

It is respectfully requested that the applicant(s) be given the benefit of the foreign filing
date(s) as evidenced by the certified papers attached hereto, in accordance with the
requirements of 35 U.S.C. § 119.

Respectfully submitted,

STAAS & HALSEY LLP

Date: July 16, 2003

By: _____


H. J. Staas
Registration No. 22,010

1201 New York Ave, N.W., Suite 700
Washington, D.C. 20005
Telephone: (202) 434-1500
Facsimile: (202) 434-1501

日 本 国 特 許 庁

JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出 願 年 月 日

Date of Application:

2002年 7月22日

出 願 番 号

Application Number:

特願2002-212387

[ST.10/C]:

[JP2002-212387]

出 願 人

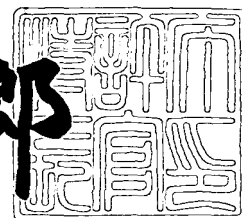
Applicant(s):

富士通株式会社

2002年10月15日

特 許 庁 長 官
Commissioner,
Japan Patent Office

太田信一郎



出証番号 出証特2002-3080944

【書類名】 特許願

【整理番号】 0251415

【提出日】 平成14年 7月22日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 09/28

【発明の名称】 並列効率計算方法

【請求項の数】 10

【発明者】

 【住所又は居所】 神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内

 【氏名】 折居 茂夫

【特許出願人】

 【識別番号】 000005223

 【氏名又は名称】 富士通株式会社

【代理人】

 【識別番号】 100103528

 【弁理士】

 【氏名又は名称】 原田 一男

 【電話番号】 045-290-2761

【手数料の表示】

 【予納台帳番号】 076762

 【納付金額】 21,000円

【提出物件の目録】

 【物件名】 明細書 1

 【物件名】 図面 1

 【物件名】 要約書 1

 【包括委任状番号】 9909129

【ブルーフの要否】 要

【書類名】 明細書

【発明の名称】 並列効率計算方法

【特許請求の範囲】

【請求項 1】

並列計算機システムの並列効率を計算する並列効率計算方法であって、

前記並列計算機システムに含まれる各プロセッサ間の負荷の均衡度合いを表すロードバランス寄与率を計算し、記憶装置に格納するロードバランス寄与率計算ステップと、

前記並列計算機システムにおいて実施した処理のうち各プロセッサにより並列計算された部分の、時間についての割合を表す仮想並列化率を計算し、記憶装置に格納する仮想並列化率計算ステップと、

前記並列計算機システムに含まれる全プロセッサの全処理時間に対する各並列性能阻害要因部分の処理時間の割合を表す並列性能阻害要因寄与率を計算し、記憶装置に格納する並列性能阻害要因寄与率計算ステップと、

前記ロードバランス寄与率と前記仮想並列化率と前記並列性能阻害要因寄与率とを用いて並列効率を計算し、記憶装置に格納するステップと、

を含む並列効率計算方法。

【請求項 2】

並列計算機システムの並列効率を計算する並列効率計算方法であって、

前記並列計算機システムに含まれる各プロセッサ間の負荷の均衡度合いを表すロードバランス寄与率を計算し、記憶装置に格納するロードバランス寄与率計算ステップと、

前記並列計算機システムにおいて実施する処理の並列化による処理時間の短縮度合いの向上の限度を表す加速率を計算し、記憶装置に格納する加速率計算ステップと、

前記並列計算機システムに含まれる全プロセッサの全処理時間に対する各並列性能阻害要因部分の処理時間の割合を表す並列性能阻害要因寄与率を計算し、記憶装置に格納する並列性能阻害要因寄与率計算ステップと、

前記ロードバランス寄与率と前記加速率と前記並列性能阻害要因寄与率とを用

いて並列効率を計算し、記憶装置に格納するステップと、
を含む並列効率計算方法。

【請求項 3】

並列計算機システムの並列効率を計算する並列効率計算方法であって、
前記並列計算機システムに含まれる各プロセッサ間の負荷の均衡度合いを表す
ロードバランス寄与率を計算し、記憶装置に格納するロードバランス寄与率計算
ステップと、

前記並列計算機システムに含まれる全プロセッサの全処理時間に対する各並列
性能阻害要因部分の処理時間の割合を表す並列性能阻害要因寄与率を計算し、記
憶装置に格納する並列性能阻害要因寄与率計算ステップと、

前記ロードバランス寄与率と前記並列性能阻害要因寄与率とを用いて並列効率
を計算し、記憶装置に格納するステップと、
を含む並列効率計算方法。

【請求項 4】

並列計算機システムの並列効率を計算する並列効率計算方法であって、
前記並列計算機システムに含まれる各プロセッサ間の負荷の均衡度合いを表す
ロードバランス寄与率を計算し、記憶装置に格納するロードバランス寄与率計算
ステップと、

前記並列計算機システムにおいて実施する処理のうち各プロセッサにより並列
計算される部分の、時間についての割合を表す仮想並列化率を計算し、記憶装置
に格納する仮想並列化率計算ステップと、

前記並列計算機システムに含まれる各プロセッサにおいて実施された処理のう
ち並列計算部分の処理時間の和と、前記各プロセッサにおいて実施された処理の
処理時間の和と、前記ロードバランス寄与率と、前記仮想並列化率とを用いて並
列効率を計算し、記憶装置に格納するステップと、

を含む並列効率計算方法。

【請求項 5】

並列計算機システムの並列効率を計算する並列効率計算方法であって、

1 プロセッサにより処理を実施する場合において当該処理のうち並列性能阻害

部分の全処理時間に相当する第 1 の処理時間を計算し、記憶装置に格納するステップと、

前記並列計算機システムに含まれる各プロセッサにおいて実施された処理のうち並列計算部分の処理時間の和である第 2 の処理時間を計算し、記憶装置に格納するステップと、

前記並列計算機システムにおいて使用したプロセッサの数と、前記並列計算機システムに含まれる各プロセッサにおいて実施された処理の処理時間のうち最長の処理時間と、前記第 1 の処理時間と、前記第 2 の処理時間とを用いて並列効率を計算し、記憶装置に格納するステップと、

を含む並列効率計算方法。

【請求項 6】

目標並列効率を設定するステップと、

計算された前記並列効率とプロセッサ数の積を前記目標並列効率で除することにより最適プロセッサ数を計算し、記憶装置に格納するステップと、

をさらに含む請求項 1 乃至 5 のいずれか 1 つ記載の並列効率計算方法。

【請求項 7】

システム増強時における増加分の稼働時間と予測並列効率とを設定するステップと、

前記並列計算機システムに現在含まれる各プロセッサにおいて実施された処理の処理時間の和と計算された前記並列効率との全処理についての積和と、前記増加分の稼働時間及び前記予測並列効率の積との和を、前記並列計算機システムに現在含まれる各プロセッサの稼働時間の和で除することにより、システム増強時の加速率を計算し、記憶装置に格納するステップと、

をさらに含む請求項 1 乃至 5 のいずれか 1 つ記載の並列効率計算方法。

【請求項 8】

前記並列計算機システムに対する新たな並列計算機システムの性能倍率を設定するステップと、

前記新たな並列計算機システムの性能倍率を用いて見積並列効率を計算し、記憶装置に格納するステップと、

をさらに含む請求項 1 乃至 5 のいずれか 1 つ記載の並列効率計算方法。

【請求項 9】

前記並列計算機システムに現在含まれる各プロセッサにおいて実施された処理の処理時間の和と計算された前記並列効率との全処理についての積和を、前記並列計算機システムに現在含まれる各プロセッサの全稼働時間で除することにより、システム運用効率を計算し、記憶装置に格納するステップと、

をさらに含む請求項 1 乃至 5 のいずれか 1 つ記載の並列効率計算方法。

【請求項 10】

並列計算機システムの並列効率を計算するためのプログラムであって、コンピュータに、

前記並列計算機システムに含まれる各プロセッサ間の負荷の均衡度合いを表すロードバランス寄与率を計算し、記憶装置に格納するロードバランス寄与率計算ステップと、

前記並列計算機システムにおいて実施した処理のうち各プロセッサにより並列計算された部分の、時間についての割合を表す仮想並列化率を計算し、記憶装置に格納する仮想並列化率計算ステップと、

前記並列計算機システムに含まれる全プロセッサの全処理時間に対する各並列性能阻害要因部分の処理時間の割合を表す並列性能阻害要因寄与率を計算し、記憶装置に格納する並列性能阻害要因寄与率計算ステップと、

前記ロードバランス寄与率と前記仮想並列化率と前記並列性能阻害要因寄与率とを用いて並列効率を計算し、記憶装置に格納するステップと、

を実行させるためのプログラム。

【発明の詳細な説明】

【0001】

【発明が属する技術分野】

本発明は、並列計算機システムの性能評価技術及び当該性能評価の結果の利用技術に関する。なお、本発明の技術は、従来のハイパフォーマンス・コンピューティング（HPC：High Performance Computing）で扱われる分野（構造解析、流体解析、計算化学等）、グリッド又はクラスタ上に展開されるバイオシミュレ

ーション、ウェブ (Web) サービス (例えばMtoM (Machine to Machine)) 等、並列処理を行う全ての分野に適用可能である。

【0002】

【従来の技術】

アプリケーション毎に並列計算機システムの性能は著しく異なる。従ってその性能評価は重要である。並列計算機システムの性能評価方法には、(1) ある特定の処理を種々の計算機で実施して比較するものと、(2) ある計算機が自己のポテンシャルに対しどの位の性能を発揮しているかを評価する自己完結型の2通りがある。前者はベンチマークテストとして主に計算機間の性能比較に使用される。後者は導入後の実運用で実施する必要がある。この自己完結型の性能評価は並列効率という指標を用いて行うことができるが、実際には実施されていない。また並列効率の代わりにプロセッサ数 p を変えながら時間測定を行い、理想的な減少の度合い $1/p$ と比較する並列性能評価 (いわゆるスケーラビリティ評価) も可能であるが、数回の時間測定を必要とするため通常は実施されない。またスケーラビリティ評価は定性的であり、厳密な並列性能評価を行うことはできない。従って現在、並列効率の悪い処理を検知できず、それらは野放し状態である。

【0003】

並列効率による並列処理の性能評価は、以下に示す式 (1) 及び (2) より決定される並列効率 $E_p(p)$ を求めることにより行われる。なお、 p はプロセッサ数、 $\tau(1)$ は1プロセッサで実行した場合の処理時間、 $\tau(p)$ は同じ処理を p 個のプロセッサで実行した場合の処理時間、 $\tau_i(p)$ は $1 \leq i \leq p$ で i 番目のプロセッサの処理時間である。

【数1】

$$E_p(p) \equiv \frac{\tau(1)}{\tau(p) \cdot p} \quad (1)$$

$$\tau(p) \equiv \max_{i=1}^p (\tau_i(p)) \quad (2)$$

例えば式 (1) は、柄谷、中村、奥田、矢川；並列有限要素法コードのGeoFEM

の性能評価, Transactions of JSCES, No.20000022 (2000)という文献に開示されている。

【0004】

【発明が解決しようとする課題】

しかし、従来の方法で並列効率を決定しても、並列性能阻害要因との定量的関係が明確でなかったため、どの阻害要因がどの位並列効率に効いているかは分からなかった。また、一部の並列性能評価技術（例えば日本国特許出願番号2001-241121、米国特許出願番号09/998160）には、図1に示すように「ロードバランスが保たれ且つ各処理時間 τ_i （並列部）、 $\alpha_{i,1}$ （冗長処理部）、 $\alpha_{i,2}$ （通信部）、 $\alpha_{i,others}$ （その他の並列性能阻害要因）が同じ」という条件が必要で、一部の並列処理にしか適用できないという問題があった。

【0005】

また、従来の方法ではグリッドやクラスタによる並列処理への適用が難しい。これはグリッドやクラスタ上に分散している、計算に必要なメモリ、データ、CPU等の資源を1つのプロセッサに集めると、1つのプロセッサで実現できない程大きな処理となる場合が多いためである。すなわち、 $\tau(1)$ を測定すること自体が難しい。また式(1)において $\tau(1)$ と $\tau(p)$ を実測で求めるということはプロセッサの性能が同じであるということを前提としているが、グリッドやクラスタ上の個々のプロセッサ性能は通常異なるため、実測した $\tau(1)$ と $\tau(p)$ を式(1)に代入しても正しい並列効率を決定できないという問題もある。

【0006】

従って本発明の目的は、「ロードバランスが保たれている」という条件をはずし、ヘテロなプロセッサ環境を含め多種類の並列処理に適用でき、並列効率と並列性能評価指標及び並列性能阻害要因間の定量的関係付けを行う並列処理性能評価技術を提供することである。

【0007】

また本発明の他の目的は、並列効率等を用いて、並列計算機システムの適切な運用を可能にするための技術を提供することである。

【0008】

さらに本発明の他の目的は、並列効率等を用いて、並列計算機システムの能力増強、更新等に対する適切な判断を可能にするための技術を提供することである。

【0009】

さらに本発明の他の目的は、並列効率等を用いて、並列計算機システムにおいて実行するプログラムのチューニングやアルゴリズムの選定を適切に実施できるようにするための技術を提供することである。

【0010】

【課題を解決するための手段】

本発明の第1の態様に係る、並列計算機システムの並列効率を計算する並列効率計算方法は、並列計算機システムに含まれる各プロセッサ間の負荷の均衡度合いを表すロードバランス寄与率 $R_b(p)$ を計算し、記憶装置に格納するロードバランス寄与率計算ステップと、並列計算機システムにおいて実施した処理のうち各プロセッサにより並列計算された部分の、時間についての割合を表す仮想並列化率 $R_p(p)$ を計算し、記憶装置に格納する仮想並列化率計算ステップと、並列計算機システムに含まれる全プロセッサの全処理時間に対する各並列性能阻害要因部分の処理時間の割合を表す並列性能阻害要因寄与率 $R_j(p)$ を計算し、記憶装置に格納する並列性能阻害要因寄与率計算ステップと、ロードバランス寄与率 $R_b(p)$ と仮想並列化率 $R_p(p)$ と並列性能阻害要因寄与率 $R_j(p)$ とを用いて並列効率を計算し（例えば実施の形態における式（4-4））、記憶装置に格納するステップとを含む。

【0011】

これにより並列効率は、ロードバランス寄与率、仮想並列化率及び並列性能阻害要因寄与率といった並列性能評価指標と定量的に関係付けられる。

【0012】

本発明の第2の態様に係る、並列計算機システムの並列効率を計算する並列効率計算方法は、並列計算機システムに含まれる各プロセッサ間の負荷の均衡度合いを表すロードバランス寄与率 $R_b(p)$ を計算し、記憶装置に格納するロードバランス寄与率計算ステップと、並列計算機システムにおいて実施する処理の並列化

による処理時間の短縮度合いの向上の限度を表す加速率 $A_p(p)$ を計算し、記憶装置に格納する加速率計算ステップと、並列計算機システムに含まれる全プロセッサの全処理時間に対する各並列性能障害要因部分の処理時間の割合を表す並列性能障害要因寄与率 $R_j(p)$ を計算し、記憶装置に格納する並列性能障害要因寄与率計算ステップと、ロードバランス寄与率 $R_b(p)$ と加速率 $A_p(p)$ と並列性能障害要因寄与率 $R_j(p)$ とを用いて並列効率を計算し（例えば実施の形態における式（４－５））、記憶装置に格納するステップとを含む。

【 0 0 1 3 】

これにより並列効率は、ロードバランス寄与率及び並列性能障害要因寄与率といった並列性能評価指標並びに加速率という補助指標と定量的に関係付けられる。

【 0 0 1 4 】

本発明の第３の態様に係る、並列計算機システムの並列効率を計算する並列効率計算方法は、並列計算機システムに含まれる各プロセッサ間の負荷の均衡度合いを表すロードバランス寄与率 $R_b(p)$ を計算し、記憶装置に格納するロードバランス寄与率計算ステップと、並列計算機システムに含まれる全プロセッサの全処理時間に対する各並列性能障害要因部分の処理時間の割合を表す並列性能障害要因寄与率 $R_j(p)$ を計算し、記憶装置に格納する並列性能障害要因寄与率計算ステップと、ロードバランス寄与率 $R_b(p)$ と並列性能障害要因寄与率 $R_j(p)$ とを用いて並列効率を計算し（例えば実施の形態における式（８－２））、記憶装置に格納するステップとを含む。

【 0 0 1 5 】

例えば、並列計算機システムに含まれる各プロセッサにおいて実施された処理のうち並列計算部分の処理時間の和が、１プロセッサにより同一処理を実施した場合の処理時間とほぼ一致する場合、すなわちほとんど並列計算できるような処理内容の場合にはこのようにして計算することができる。

【 0 0 1 6 】

本発明の第４の態様に係る、並列計算機システムの並列効率を計算する並列効率計算方法は、並列計算機システムに含まれる各プロセッサ間の負荷の均衡度合

いを表すロードバランス寄与率 $Rb(p)$ を計算し、記憶装置に格納するロードバランス寄与率計算ステップと、並列計算機システムにおいて実施する処理のうち各プロセッサにより並列計算される部分の、時間についての割合を表す仮想並列化率 $Rp(p)$ を計算し、記憶装置に格納する仮想並列化率計算ステップと、並列計算機システムに含まれる各プロセッサにおいて実施された処理のうち並列計算部分の処理時間の和と、各プロセッサにおいて実施された処理の処理時間の和と、ロードバランス寄与率 $Rb(p)$ と、仮想並列化率 $Rp(p)$ とを用いて並列効率を計算し（例えば実施の形態における式（9-1））、記憶装置に格納するステップとを含む。本発明の第1の態様の変形例である。

【0017】

本発明の第5の態様に係る、並列計算機システムの並列効率を計算する並列効率計算方法は、1プロセッサにより処理を実施する場合において当該処理のうち並列性能阻害部分の全処理時間に相当する第1の処理時間を計算し、記憶装置に格納するステップと、並列計算機システムに含まれる各プロセッサにおいて実施された処理のうち並列計算部分の処理時間の和である第2の処理時間を計算し、記憶装置に格納するステップと、並列計算機システムにおいて使用したプロセッサの数と、並列計算機システムに含まれる各プロセッサにおいて実施された処理の処理時間のうち最長の処理時間と、第1の処理時間と、第2の処理時間とを用いて並列効率を計算し（例えば実施の形態における式（9-2））、記憶装置に格納するステップとを含む。

【0018】

所定のモデル化に基づき一度の測定により得られる処理時間だけで並列効率が計算できるようになっている。

【0019】

また、上で述べたロードバランス寄与率計算ステップにおいて、上記ロードバランス寄与率 $Rb(p)$ を、並列計算機システムに含まれる全プロセッサにおいて実施された処理の全処理時間を、並列計算機システムに含まれる各プロセッサにおいて実施された処理の処理時間のうち最長の処理時間及び並列計算機システムにおいて使用したプロセッサ数により除することにより計算する（例えば実施の形

態における式 (5)) ような構成とすることも可能である。

【 0 0 2 0 】

さらに、上で述べた仮想並列化率計算ステップにおいて、仮想並列化率 $R_p(p)$ を、並列計算機システムに含まれる各プロセッサにおいて実施された処理のうち並列計算部分の処理時間の和を、1 プロセッサにより同一処理を実施した場合の第3の処理時間に相当する処理時間により除することにより計算する（例えば実施の形態における式 (6-1)) ような構成とすることも可能である。

【 0 0 2 1 】

また、上で述べた並列性能阻害要因寄与率計算ステップにおいて、特定の並列性能阻害要因についての並列性能阻害要因寄与率 $R_j(p)$ を、並列計算機システムに含まれる各プロセッサにおける特定の並列性能阻害要因部分の処理時間の和を、並列計算機システムに含まれる各プロセッサの処理時間の和により除することにより計算する（例えば実施の形態における式 (7)) ような構成とすることも可能である。

【 0 0 2 2 】

また、上で述べた加速率計算ステップにおいて、上記加速率を、並列計算機システムに含まれる各プロセッサにおいて実施された処理のうち並列計算部分の処理時間の和を1 プロセッサにより同一処理を実施した場合の第3の処理時間に相当する処理時間により除することにより計算される仮想並列化率を、1 から差し引いた値の逆数として計算する（例えば実施の形態における式 (6-2)) ような構成とすることも可能である。

【 0 0 2 3 】

さらに、上で述べた処理時間は、実際の処理時間に加え、対応する事象の確認回数で表される場合もある。

【 0 0 2 4 】

また、計算された並列効率に並列計算機システムにおいて使用したプロセッサ数を乗じて補助指標を計算し、記憶装置に格納するステップをさらに含むような構成であってもよい。これにより、並列計算機システムにおいてプロセッサ何個分の処理を実施したかを提示することができるようになる。

【 0 0 2 5 】

さらに、上で述べた第 3 の処理時間を、1 プロセッサにより処理を実施する場合において当該処理のうち並列性能阻害部分の全処理時間に相当する第 1 の処理時間と並列計算機システムに含まれる各プロセッサにおいて実施された処理のうち並列計算部分の処理時間の和である第 2 の処理時間との和により計算する（例えば実施の形態における式（15））ような構成であってもよい。所定のモデル化により処理時間の 1 度の測定にて並列効率等の計算が可能になる。

【 0 0 2 6 】

さらに、上で述べた第 1 の処理時間が、並列計算機システムに含まれる各プロセッサにおいて実施された処理のうち冗長処理又は通信処理の処理時間の和により計算される（例えば実施の形態における式（12-1）等）ような構成とすることも可能である。

【 0 0 2 7 】

また、本発明の第 1 乃至第 5 の態様において、目標並列効率を設定するステップと、計算された並列効率とプロセッサ数の積を目標並列効率で除することにより最適プロセッサ数を計算し、記憶装置に格納するステップとをさらに含むような構成も可能である。多くのプロセッサを投入しても処理時間の短縮につながるとは限らず、このように最適プロセッサ数が計算できれば無駄な資源の投入を防止することができる。

【 0 0 2 8 】

さらに、本発明の第 1 乃至第 5 の態様において、システム増強時における増加分の稼働時間と予測並列効率とを設定するステップと、並列計算機システムに現在含まれる各プロセッサにおいて実施された処理の処理時間の和と計算された並列効率との全処理についての積和と、増加分の稼働時間及び予測並列効率の積との和を、並列計算機システムに現在含まれる各プロセッサの稼働時間の和で除することにより、システム増強時の加速率を計算し（例えば実施の形態における式（18））、記憶装置に格納するステップとをさらに含むような構成であってもよい。システム増強時においてシステム運用者に適切な定量的指針を与えることができるようになる。

【 0 0 2 9 】

また、本発明の第 1 乃至第 5 の態様において、並列計算機システムに対する新たな並列計算機システムの性能倍率を設定するステップと、新たな並列計算機システムの性能倍率を用いて見積並列効率を計算し、記憶装置に格納するステップとをさらに含むような構成であってよい。システムリプレイス時における定量的指針を与えることができるようになる。

【 0 0 3 0 】

さらに、本発明の第 1 乃至第 5 の態様において、並列計算機システムに現在含まれる各プロセッサにおいて実施された処理の処理時間の和と計算された並列効率との全処理についての積和を、並列計算機システムに現在含まれる各プロセッサの稼働時間の和で除することにより、システム運用効率を計算し（例えば実施の形態における式（17））、記憶装置に格納するステップとをさらに含むような構成であってよい。従来の稼働率という考え方に比して本発明のように並列効率を考慮したシステム運用効率を使用した方が、システム運用状況をより実際に即した形で評価することができるようになる。

【 0 0 3 1 】

また、本発明の第 1 乃至第 5 の態様において、目標処理時間を設定するステップと、目標処理時間を用いて目標並列効率を計算し、記憶装置に格納するステップと、目標並列効率の妥当性を確認するステップとをさらに含むような構成であってもよい。例えば目標並列効率は、線形外挿にて計算することができる。

【 0 0 3 2 】

さらに、目標並列効率の妥当性が確認された場合には、チューニング実施後の並列効率を計算し、記憶装置に格納するステップと、チューニング実施後の並列効率と目標並列効率とを比較するステップとをさらに含むような構成であってもよい。より定量的な観点で、アプリケーション等のチューニングを実施することができるようになる。

【 0 0 3 3 】

また、本発明の第 1 乃至第 5 の態様において、目標処理時間を設定するステップと、異なるアルゴリズム毎に当該アルゴリズムにおける並列効率を用いて必要

となるプロセッサ数の見積値を計算し、記憶装置に格納するステップと、プロセッサ数の見積値が並列計算機システムにおいて実施する当該アルゴリズムによる処理の並列化による処理時間の短縮度合いの向上の限度を表す加速率より小さく、異なるアルゴリズムについて計算されたプロセッサ数の見積値のうち最小の値となるアルゴリズムを抽出するステップとをさらに含むような構成であってもよい。より並列効率を向上させることができるアルゴリズムを定量的に選択することができるようになる。

【 0 0 3 4 】

なお、本発明に係る並列効率計算方法はプログラム及びコンピュータにて実施することができ、当該プログラムをコンピュータで実行する場合には当該コンピュータは並列効率計算装置となる。また、このようなプログラムは、例えばフレキシブルディスク、CD-ROM、光磁気ディスク、半導体メモリ、ハードディスク等の記憶媒体又は記憶装置に格納される。また、ネットワークなどを介して配布される場合もある。尚、中間的な処理結果はメモリに一時保管される。

【 0 0 3 5 】

【発明の実施の形態】

[本発明の原理]

本発明では、並列効率 $E_p(p)$ を並列性能評価指標で記述することにより、並列効率 $E_p(p)$ を並列性能阻害要因と定量的に結び付ける。図2のように、並列処理時間 $\tau_i(p)$ は、並列計算部の処理時間 $\gamma_i(p)$ と、各並列性能阻害要因 j の処理時間 $\chi_{i,j}(p)$ との和で式(3)のように表わすことができる。ここで $1 \leq j \leq j_{others}$ である。なお、図2において i はプロセッサ番号であり、 p はプロセッサ個数である。また、図2ではプロセッサ i とプロセッサ $i+1$ についてのみ示されている。

【数2】

$$\tau_i(p) \equiv \gamma_i(p) + \sum_{j=1}^{j_{others}} \chi_{i,j}(p) \quad (3)$$

【 0 0 3 6 】

そして式(1)を以下のように変形し、さらに並列性能評価指標であるロード
バランス寄与率 $Rb(p)$ 、仮想並列化率 $Rp(p)$ 、並列性能阻害要因寄与率 $Rj(p)$ を
導入して並列効率 $E_p(p)$ を記述する。

【数 3】

$$E_p(p) = \frac{\tau(1)}{\tau(p) \cdot p} \cdot \frac{\sum_{i=1}^p \tau_i(p)}{\sum_{i=1}^p \tau_i(p)} \quad (4-1)$$

$$= \frac{\sum_{i=1}^p \tau_i(p)}{\tau(p) \cdot p} \cdot \frac{\tau(1)}{\sum_{i=1}^p \tau_i(p)} \cdot \frac{\sum_{i=1}^p \gamma_i(p)}{\sum_{i=1}^p \gamma_i(p)} \quad (4-2)$$

$$= \frac{\sum_{i=1}^p \tau_i(p)}{\tau(p) \cdot p} \cdot \frac{\tau(1)}{\sum_{i=1}^p \gamma_i(p)} \cdot \frac{\sum_{i=1}^p \gamma_i(p)}{\sum_{i=1}^p \tau_i(p)} \quad (4-3)$$

【0 0 3 7】

なお、式(1)から式(4-1)への変形は、式(1)の分子及び分母に $\tau_i(p)$ の i についての和を掛け算することにより行われる。また、式(4-2)への変形は、式(4-1)における各要素の位置を変更すると共に、式(4-1)の分子及び分母に $\gamma_i(p)$ の i についての和を掛け算することにより行われる。また、以下の式が式(3)から導かれる。これは、 $\tau_i(p)$ の i についての和を表すものである。

【数 4】

$$\sum_{i=1}^p \tau_i(p) = \sum_{i=1}^p \gamma_i(p) + \sum_{j=1}^{j_{Others}} \sum_{i=1}^p \chi_{i,j}(p)$$

【0038】

そうすると、ロードバランス寄与率 $R_b(p)$ 、仮想並列化率 $R_p(p)$ 、並列性能阻害要因寄与率 $R_j(p)$ により以下の式が導かれる。

【数 5】

$$E_p(p) = R_b(p) \cdot \frac{1}{R_p(p)} \cdot \left(1 - \sum_{j=1}^{j_{Others}} R_j(p) \right) \quad (4-4)$$

また加速率 $A_p(p)$ を用いると以下のようにも表される。

【数 6】

$$E_p(p) = R_b(p) \cdot \frac{1}{1 - 1/A_p(p)} \cdot \left(1 - \sum_{j=1}^{j_{Others}} R_j(p) \right) \quad (4-5)$$

【0039】

なお、ロードバランス寄与率 $R_b(p)$ 、仮想並列化率 $R_p(p)$ 、並列性能阻害要因寄与率 $R_j(p)$ 、加速率 $A_p(p)$ は以下のように表される。

【数 7】

$$R_b(p) \equiv \frac{\sum_{i=1}^p \tau_i(p)}{\tau(p) \cdot p} \quad (1/p \leq R_b(p) \leq 1) \quad (5)$$

$$R_p(p) \equiv \frac{\sum_{i=1}^p \gamma_i(p)}{\tau(1)} \quad (0 \leq R_p(p) \leq 1) \quad (6-1)$$

$$A_p(p) = \frac{1}{1 - R_p(p)} \quad (6-2)$$

$$R_j(p) \equiv \frac{\sum_{i=1}^p \chi_{i,j}(p)}{\sum_{i=1}^p \tau_i(p)} \quad (0 \leq R_j(p) \leq 1) \quad (7)$$

なお、並列性能阻害要因は j で番号付けされる。

【0 0 4 0】

ロードバランスが保たれた状態とは、図 1 で示したように、プロセッサの処理時間 $\tau_i(p)$ が均等の状態である。式 (5) はこの状態を $R_b(p) = 1$ とし、更に保たれない状態を $1/p \leq R_b(p) \leq 1$ で表す。図 3 のように並列処理時に 1 台のプロセッサのみで処理する場合には式 (5) の分子が $\tau(p)$ となるため、ロードバランス寄与率 $R_b(p)$ は $1/p$ の下限値となる。また式 (5) によれば、ロードバランス寄与率 $R_b(p)$ は並列効率 $E_p(p)$ の比率となり、並列性能を直感的に把握することを容易にする。

【0 0 4 1】

仮想並列化率 $R_p(p)$ は並列計算部の処理時間 γ_i の和が $\tau_1(1)$ に占める割合である。これが 1 より小さい場合、その処理は並列処理できない処理を含んでいることを示す。この割合により並列性能の上限を加速率 $A_p(p)$ として表わすことが

できる。 $A_p(p)$ は、プロセッサを無限に投入した時の理想的な上限値 $A_p(p) = \tau(1) / \sum_j \chi_{1,j}(1) = 1 / (1 - (\sum_i \gamma_i(p) / \tau(1)))$ である。通常の $\tau(1) / \tau(p)$ は、並列性能阻害要因により $A_p(p)$ より小さな値となる。

【0042】

並列性能阻害要因寄与率 $R_j(p)$ は式(7)で示すように $\tau_i(p)$ の i についての和で規格化されているため、高並列、低並列に関わらず並列性能阻害要因の寄与を処理時間の割合として把握できるようになっている。またこの割合が並列効率の比率となるため、並列性能の阻害を定量的に把握することができる。

【0043】

式(2)，(5)，(6-1)，(7)の各変数は、 $\tau(1)$ を除く全てが並列実行時に測定可能である。式(8-1)が成り立つ時、式(6-1)から仮想並列化率 $R_p(p)$ はほぼ1となり、式(4-4)は式(8-2)に等価となる。

【数8】

$$\tau(1) \cong \sum_{i=1}^p \gamma_i(p) \quad (8-1)$$

$$E_p(p) = R_b(p) \cdot \left(1 - \sum_{j=1}^{j_{Others}} R_j(p) \right) \quad (8-2)$$

すなわち、並列効率 $E_p(p)$ は $\tau(1)$ という推定値を用いずに済むので、正確に決定することができる。一方、条件式(8-1)に関係なく、式(8-2)を、式(4-4)，(4-5)の代替値として並列性能評価において用いることも可能である。この場合、式(8-2)の値は $R_p(p) \leq 1$ ゆえ、式(4-4)，(4-5)の値に等しいか小さい値となる。

【0044】

並列効率 $E_p(p)$ は、上で述べた式(4-4)，(4-5)，(8-2)及び次の式(9-1)でも計算することが出来る。

【数 9】

$$E_p(p) = R_b(p) \cdot \frac{1}{R_p(p)} \cdot \frac{\sum_{i=1}^p \gamma_i(p)}{\sum_{i=1}^p \tau_i(p)} \quad (9-1)$$

式 (9-1) は、式 (4-3) からロードバランス寄与率 $R_b(p)$ 及び仮想並列化率 $R_p(p)$ のみを用いて変形した結果である。

【0 0 4 5】

また、式 (3) より $\tau(1)$ は式 (10) となる。

【数 10】

$$\tau_1(1) = \gamma_1(1) + \sum_{j=1}^{j_{Others}} \chi_{1,j}(1) \quad (10)$$

ここで $\gamma_1(1)$ と $\chi_{1,j}(1)$ を、 $\gamma_i(p)$ と $\chi_{i,j}(p)$ を用いてモデル化する。グリッドやクラスタのように異なった CPU 性能を有する計算機による並列処理での $\tau_1(1)$ の実測は不可能であるため、このモデル化により $\tau(1)$ が決定でき、式 (6-1) の仮想並列化率 $R_p(p)$ を計算することが可能となる。

【0 0 4 6】

プロセッサ性能が同じで理想的な場合、並列計算部を p 個のプロセッサで処理すると、処理時間は $p=1$ と比較して $1/p$ となる。この場合、 $\tau_i(p) = \gamma_i(p)$ であり、任意のプロセッサの $\gamma_i(p)$ を p 倍すると $\gamma_1(1)$ を求めることができる。一方グリッドやクラスタ上には異なった CPU 性能を有する計算機が存在するのが通常であり、 p 個のプロセッサにおいて実測された $\gamma_i(p)$ を基に、プロセッサ数 1 の場合の $\gamma_1(1)$ を式 (11) のように推定する。

【数 11】

$$\gamma_1(1) \equiv \sum_{i=1}^p \gamma_i(p) \quad (11)$$

この式 (11) の概念図を図 4 に示す。式 (11) のモデル化により、個々のプロセッサの性能が異なっている場合でも、仮想的に 1 プロセッサの $\gamma_1(1)$ の時間を決めることができる。

【0047】

また、 $\chi_{1,j}(1)$ を冗長処理とそれ以外という 2 つに分けてモデル化する。 $\gamma_1(1)$ に属さない処理時間はすべて $\chi_{1,j}(1)$ に含まれるものとする。

【0048】

(1) 冗長処理時間のモデル化

各プロセッサが全く同じ処理を行うとき、ここではそれを冗長処理と呼ぶ。この処理は並列処理ではなく、プロセッサが増えても処理時間は減少しない。そこで $j = 1$ を冗長処理とし、その時間 $\chi_{1,1}(1)$ を式 (12-1) 乃至 (12-4) のようにモデル化する。

【数 12】

$$\chi_{1,1}(1) \equiv \frac{1}{p} \cdot \sum_{i=1}^p \chi_{i,1}(p) \quad (12-1)$$

$$\chi_{1,1}(1) \equiv \text{Max}_{i=1}^p (\chi_{i,1}(p)) \quad (12-2)$$

$$\chi_{1,1}(1) \equiv \text{Min}_{i=1}^p (\chi_{i,1}(p)) \quad (12-3)$$

$$\chi_{1,1}(1) \equiv \chi_{ii,1}(p) \quad (12-4)$$

ここで ii は以下の式における ii である。

【数 13】

$$\tau_{ii}(p) \equiv \text{Max}_{i=1}^p \left(\gamma_i(p) + \sum_{j=1}^{j_{\text{others}}} \chi_{i,j}(p) \right)$$

【0049】

冗長処理は、同じ手続き (処理内容) ではあるがデータが異なる処理を並列処

理する、いわゆるデータパラレルに多い処理である。データパラレルの場合、ロードバランスを保つため、CPU性能が同じプロセッサによる並列処理であることを想定している。そこでプロセッサ毎の冗長処理時間の違いを各プロセッサの時間測定に起因するばらつきによるものと考えることができる。この場合、各プロセッサの測定値を平均化した式(12-1)を適用するのが妥当である。

【0050】

一方グリッドやクラスタでは異なる性能を有するプロセッサが紛れ込むことも想定される。この異なる性能を有するプロセッサによる影響を的確に捉えようとする場合、式(12-2)及び(12-3)を用いる。式(12-3)を用いれば式(6-1)の仮想並列化率 $R_p(p)$ は最小に見積もられ、並列効率 $E_p(p)$ は最大となる。式(12-3)を用いれば仮想並列化率 $R_p(p)$ は最大に見積もられ、並列効率 $E_p(p)$ は最小となる。これら2つの並列効率 $E_p(p)$ を比べて、異なるCPU性能のプロセッサでデータパラレル処理を行っていることの検知が可能となる。

【0051】

$\tau(p)$ は式(2)で決まり、そのプロセッサ i の冗長処理時間は式(12-4)の値である。従って、並列効率 $E_p(p)$ を決定したデータを解析すると考えれば式(12-4)の使用が妥当である。一方、この式はプロセッサ i の情報のみから $x_{1,1}(1)$ が決定されることを意味し、他のプロセッサの値と大きく異なる場合を検知できないという欠点を持つ。この例として図5はCPU性能が1/5のプロセッサがプロセッサ1($i=1$)である場合を示す。式(12-4)では冗長処理の時間をプロセッサ1の値のみで評価することになる。式(12-1)では4プロセッサの各値の平均で評価する。式(12-2)ではプロセッサ1の値で、式(12-3)では $i=2, 3, 4$ のプロセッサの値で評価することになる。従って式(12-1)をベースとして、必要に応じて他の定義を使用するのが妥当である。

【0052】

(2) $x_{1,j}(1)$ ($2 \leq j \leq j_{\text{others}}$) のモデル化

実際に並列性能阻害要因による処理時間を測定すると、 $x_{1,j}(1) \neq 0$ の場合が

存在する。この処理時間は並列処理しても減らないため、式(6-1)の仮想並列化率 $R_p(p)$ に反映され、式(6-2)の加速率 A_p が有限の値となり、処理に投入して意味があるプロセッサ数の上限値が決まる。そこで冗長処理以外の並列性能阻害要因による処理時間 $\chi_{1,j}(1)$ ($2 \leq j \leq j_{\text{others}}$) を、 $p=1$ の処理時間 $\chi_{1,j}(1)$ と $p>1$ の処理時間を表わす式(13-2)で式(13-1)のようにモデル化し、 $\chi_{i,j}(p)$ と $p>1$ で発生し且つ p に依存する並列性能阻害要因による処理時間 $X_{i,j}(p)$ を測定して $\chi_{1,j}(1)$ を求める。すなわち、 $\chi_{1,j}(1) = \chi_{i,j}(p) - X_{i,j}(p)$ であり、右辺の2項は両方とも測定により求めるものとする。

【数 1 4】

$$\chi_{i,j}(p) \equiv X_{i,j}(p) + \chi_{1,j}(1) \quad (13-1)$$

$$X_{1,j}(1) = 0 \quad (13-2)$$

【0 0 5 3】

例として $\chi_{1,j}(1) \neq 0$ の場合で $p=1$ の処理時間を図6(a)に、 $p=4$ の処理時間を図6(b)に示す。図6(b)に示されるように、並列処理時の $\chi_{i,2}(p)$ は $p=1$ の処理時間 $\chi_{1,2}(1)$ に $X_{i,2}(p)$ をプラスした式(13-1)のようになる。このような現象は、通信等で通信のハードウェアを起動するまでの前処理が $p=1$ の時にも実行される場合等で観測される。

【0 0 5 4】

式(13-1)及び(13-2)より $\chi_{1,j}(1)$ ($2 \leq j \leq j_{\text{others}}$) は冗長処理と同様に式(13-3), (13-4), (13-5)により求めることができる。

【数 1 5】

$$\chi_{1,j}(1) \equiv \frac{1}{p} \cdot \sum_{i=1}^p (\chi_{i,j}(p) - X_{i,j}(p)) \quad (13-3)$$

$$\chi_{1,j}(1) \equiv \text{Max}_{i=1}^p (\chi_{i,j}(p) - X_{i,j}(p)) \quad (13-4)$$

$$\chi_{1,j}(1) \equiv \text{Min}_{i=1}^p (\chi_{i,j}(p) - X_{i,j}(p)) \quad (13-5)$$

例えば図 6 (b) では $\chi_{i,2}(p) = \chi_{1,2}(1) + X_{i,2}(p)$ と実測されるので、 $X_{i,2}(p) = 5, 6, 7, 8$ を実測して式 (13-3), (13-4), (13-5) から $\chi_{1,2}(1) = 5$ を算出できる。この値は図 6 (a) の $\chi_{1,2}(1)$ と一致する。

【0 0 5 5】

(3) $\chi_{i,j\text{others}}(p)$ の決定方法

並列処理障害要因に分類して実測できない $\chi_{i,j\text{others}}(p)$ は式 (13-6) で求める。

【数 1 6】

$$\chi_{i,j\text{others}}(p) = \tau_i(p) - \gamma_i(p) - \sum_{j=1}^{j\text{others}-1} \chi_{i,j}(p) \quad (13-6)$$

【0 0 5 6】

次に、モデル化により求められた式 (11) により、式 (10) を以下の式で書き直す。

【数 1 7】

$$\tau(1) = \sum_{i=1}^p \gamma_i(p) + \sum_{j=1}^{j\text{others}} \chi_{1,j}(1)$$

また、式 (8-1) は、以下のように変形される。

【数 1 8】

$$\sum_{i=1}^p \gamma_i(p) + \sum_{j=1}^{j_{Others}} \chi_{1,j}(1) \cong \sum_{i=1}^p \gamma_i(p)$$

この式が成り立つためには式 (14) の条件を満たす必要がある。この条件式は測定値より求めた値の大小関係の比較であるため、具体的に判定を行うことができる。

【数 1 9】

$$\sum_{i=1}^p \gamma_i(p) \gg \sum_{j=1}^{j_{Others}} \chi_{1,j}(1) \quad (14)$$

【0 0 5 7】

この式 (14) の $\gamma_i(p)$ は実測により求まる。また $\chi_{1,j}(1)$ の j についての和は、モデル化式 (12-1), (12-2), (12-3), (12-4) 及び式 (13-1), (13-2), (13-3), (13-4), (13-5) より求めることができる。その結果、初めて式 (8-1) の判定が式 (14) を用いて具体的に可能となる。例えば以下のような場合、式 (14) の条件が成立し、式 (6-1) の仮想並列化率 $R_p(p)$ はほぼ 1 となり、式 (4-4), (4-5) は式 (8-2) と等価となり、推定値である $\tau(1)$ の影響がほぼゼロになると言う意味で正確な並列効率 $E_p(p)$ を得ることができる。

【数 2 0】

$$\sum_{i=1}^p \gamma_i(p) = 1000, \quad \sum_{j=1}^{j_{Others}} \chi_{1,j}(1) = 10$$

【0 0 5 8】

また式 (10), (11), (12-1), (12-2), (12-3), (12-4), (13-1), (13-2), (13-3), (13-4), (13-5), (13-6) により $\tau(1)$ も具体的に計算することができる式 (15) となる。式 (15) は、 $\tau(1)$ を各プロセッサの並列処理時間 $\gamma_i(p)$ の総和と

$p = 1$ の時の並列効率阻害要因による処理時間 $\chi_{1,j}(1)$ の和として表わしたものである。

【数 2 1】

$$\tau(1) = \sum_{i=1}^p \gamma_i(p) + \sum_{j=1}^{j_{Others}} \chi_{1,j}(1) \quad (15)$$

【0 0 5 9】

以上説明した式 (1), (2), (15) より以下に示す式 (9-2) を得る。

【数 2 2】

$$E_p(p) \equiv \frac{\sum_{j=1}^{j_{Others}} \chi_{1,j}(1) + \sum_{i=1}^p \gamma_i(p)}{\tau(p) \cdot p} \quad (9-2)$$

【0 0 6 0】

式 (9-1) 及び (9-2) は、並列計算部の時間 $\gamma_i(p)$ の i についての和を用いるもので、式 (4-4), (4-5) と比べ、 $\chi_{i,j}(p)$ のデータなしに並列効率 $E_p(p)$ が求まる利点がある。但し、 $\chi_{1,j}(1)$ のデータは必要である。

【0 0 6 1】

式 (4-4), (4-5), (7) で示したように、本発明では並列性能阻害要因 j は任意の数を追加できる。並列性能阻害要因の追加例を図 7 (a) 及び図 7 (b) に示す。図 7 (b) は立ち上がり時間 χ_{TC} を考慮して時間測定した場合、図 7 (a) は同じ処理で時間測定しない場合の処理時間を示す。図 7 (a) の場合には、以下のような計算がなされる。

$$\tau_1 = 10 + 5 + 90 + 20 + 20 = 145$$

$$\tau_2 = 10 + 80 + 10 = 100$$

$$\tau_3 = 15 + 80 + 10 = 105$$

$$\tau_4 = 10 + 90 + 10 = 110$$

$$Rb(4) = (145 + 100 + 105 + 110) / (145 \times 4) = 0.7931$$

$$R_C(4) = (25+20+25+20)/460 = 0.1957$$

$$R_P(4) = 1 \text{ (仮定)}$$

$$E_P(4) = 0.7931 \times 1 \times (1 - 0.1957) = 0.6379$$

【 0 0 6 2 】

また、図 7 (b) の場合には、以下のような計算がなされる。

$$\tau_1 = 10 + 5 + 90 + 20 + 20 = 145$$

$$\tau_2 = 5 + 10 + 80 + 10 = 105$$

$$\tau_3 = 10 + 15 + 80 + 10 = 115$$

$$\tau_4 = 15 + 10 + 90 + 10 = 125$$

$$R_b(4) = (145+105+115+125)/(145 \times 4) = 0.8448$$

$$R_C(4) = (25+20+25+20)/490 = 0.1837$$

$$R_{TC}(4) = (0+5+10+15)/490 = 0.0612$$

$$R_P(4) = 1 \text{ (仮定)}$$

$$E_P(4) = 0.8448 \times 1 \times (1 - 0.1837 - 0.0612) = 0.6379$$

【 0 0 6 3 】

これらの並列性能評価指標の値を図 8 にまとめて示す。図 7 (a) から求めた値(ケース 1) と比べると、図 7 (b) から求めた値(ケース 2) は、立ち上がり時間のための R_{TC} を追加することにより R_C が減少し R_b が増加するが、 E_P は同じであることが分かる。この場合、並列性能阻害要因を追加することにより E_P が変わるのではなく、その内訳がより明確になる。

【 0 0 6 4 】

式 (5) で示したようにロードバランス寄与率 $R_b(p)$ を表現することにより、ロードバランスと並列効率 $E_P(p)$ を関係付けることができる。ロードバランス寄与率 $R_b(p)$ を式 (5) のように定義する理由は、図 9 に示すように並列性能阻害要因の寄与が各プロセッサにおいて異なった状態でロードバランスが成り立つ場合を考慮できるからである。図 9 では、例えばプロセッサ 1 の並列処理部分は他に比べ非常に少なく、冗長処理が非常に多くなっているが、全てのプロセッサの処理時間は一致しているのでロードバランスは保たれている。すなわち、 $\gamma_i(p)$ と $\chi_{i,j}(p)$ とが個々にはバランスしていないが全体にはバランスしているという

状態である。なお、 $x_{i,j\text{others}}(p) (= \tau_i - \gamma_i - x_{i,1} - x_{i,2})$ は例えば I / O による処理時間である。

【0065】

図9の場合、ロードバランス寄与率 $R_b(p)$ は1である。図10のように並列処理において1台のプロセッサのみで処理する場合、 $R_b(p)$ は下限 $1/p$ となる。また、図11のように、プロセッサ1とプロセッサ2の処理時間は一致しているが、プロセッサ3及び4の処理時間とは一致しておらず、ロードバランスが取れていない。この場合に、 $R_b(p)$ は以下のとおりになる。

【数23】

$$R_b(p) = \frac{\sum_{i=1}^p \tau_i(p)}{\tau(p) \cdot p} = \frac{100 + 100 + 80 + 70}{100 \times 4} = 0.8750$$

【0066】

さらに、低並列のときは顕在化しなかった並列処理阻害要因が高並列で顕在化する場合がある。従来の性能評価指標の1つである並列化率（＝（ $p=1$ の並列処理部の処理時間）／（（ $p=1$ の並列処理部の処理時間）＋（ $p=1$ の並列処理できない部分の処理時間）））では、この現象を十分に捉えることができなかった。例えば図12の例では、 $p=1$ における並列化率は0.99（＝198／（198＋2））であり、残りの0.01が並列処理できない処理時間の割合である。ところがこの値は図中の $p=100$ の場合のような高並列でも2時間であり、並列処理できない部分が50％（＝2／（1.98＋2））を占める現実を反映していない。本発明では、式（7）で示したように、並列性能阻害要因 $R_j(p)$ を、 $x_{i,j}(p)$ の i についての和を $\tau_i(p)$ の i についての和で規格化した値として表現している。この規格化により、高並列で $\tau(p)$ が小さな値になった時も、 $R_j(p)$ の上限は1となり、各々の並列性能阻害要因の影響を並列処理時の百分率で表わすことができる。

【0067】

以上述べたように、並列効率 $E_p(p)$ を計算すると共に、並列性能評価指標 $R_b(p)$

p), $R_p(p)$, $R_j(p)$ ($R_{RED}(4)$, $R_C(4)$, . . . , $R_{Others}(4)$) と補助指標 $A_p(p)$, $E_p(p) \cdot p$ をも計算することができる。この計算結果の一例を図 1 3 に示す。この図 1 3 に示した 8 つの項目で、並列性能を定量的に表現することができる。

【 0 0 6 8 】

図 1 3 に示したように、 $E_p(p) \cdot p = 1.777$ であるから、4 プロセッサ構成の並列計算機システムであるが 1.777 プロセッサの性能で処理していることが分かる。並列効率はロードバランス寄与率で 94% ($R_b(4) = 0.9392$) に低下する。並列性能阻害要因の影響は、冗長処理が 22% ($R_{RED}(4) = 0.2230$)、通信が 33% ($R_C(4) = 0.3309$)、その他が 3% ($R_{Others}(4) = 0.0288$) である。従って通信と冗長処理で 55% 並列性能を低下させている。図 1 3 では $R_p(4) = 0.8821$ ということから、プロセッサを無限に投入した時の並列最大性能が 1 プロセッサの 8.482 ($= A_p(4) = 1 / (1 - 0.8821)$) 倍であることが推定できる。従ってこの処理は 8 プロセッサ以下で行われるべき処理であることが分かる。

【 0 0 6 9 】

また、例えば並列効率の設定目標値 (E_p)_T を 0.8 と設定した場合、図 1 3 で示されるような処理群を想定すると、最適なプロセッサ数は以下の式で計算される。

$$\begin{aligned} (p)_{OPT} &= E_p(4) / (E_p)_T \cdot p \\ &= 0.4443 / 0.8 \times 4 = 2.215 \end{aligned}$$

従って、最適なプロセッサの見積値は $(p)_{OPT} = 2$ となる。

【 0 0 7 0 】

なお処理群とは、同一アプリケーション・プログラムで同じ機能を使い入力データだけを変えた複数の処理のことであり、科学技術計算のパラメトリック・スタディ等で頻繁に実施される処理である。

【 0 0 7 1 】

従来、並列計算機システムの評価は、以下で示す式 (16) に示す稼働率 (Net Working Rate) NWR_{system} で行われていた。しかし、並列効率の低い処理も含

まれる場合があるため、稼働率がよいからといって必ずしもシステムの運用効率が高いとは限らなかった。

【数 2 4】

$$NWR_{System} \equiv \frac{\sum_{k=1}^{K_{\max}} \left[\sum_{i=1}^p \tau_i(p) \right]_k}{P_{System} \sum_{i=1} T_i} \quad (16)$$

稼働時間と処理時間の総和（以下の式）の例を図 1 4 に示す。

【数 2 5】

$$\sum_{k=1}^{K_{\max}} \left[\sum_{i=1}^p \tau_i(p) \right]_k$$

図 1 4 では、稼働時間 T_i に対して処理時間の総和は少なくなっている。この減少の度合いはプロセッサによって異なる。

【0 0 7 2】

本発明により、式（16）を基にシステム運用効率 E_{system} という指標を作り、システムの運用効率を評価することが可能となる。この指標向上のためにはどの処理の並列効率がどの位向上する必要があるか等、運用効率の向上に対して具体的指針を出すことが可能となる。

【数 2 6】

$$E_{System} \equiv \frac{\sum_{k=1}^{K_{\max}} \left[E_p(p) \cdot \sum_{i=1}^p \tau_i(p) \right]_k}{P_{System} \sum_{i=1} T_i} \quad (17)$$

【0 0 7 3】

例えば $P_{\text{system}} = 4$ 、 $T_i = 10$ 、 $k_{\text{max}} = 2$ であり、さらに以下の条件が満たされるとすると、 $E_{\text{system}} = (5 + 9) / (10 + 10 + 10 + 10) = 0.35$ となる。

【数 2 7】

$$\left[E_p(p) \cdot \sum_{i=1}^p \tau_i(p) \right]_1 = 0.5 \times (4 + 3 + 2 + 1) = 5$$

$$\left[E_p(p) \cdot \sum_{i=1}^p \tau_i(p) \right]_2 = 1 \times (9) = 9$$

なお、従来の稼働率 $NWR_{\text{system}} = (10 + 9) / 40 = 0.4838$ となる。並列効率を考慮することにより、並列処理により各処理において無駄にしている時間を考慮したシステムの運用効率を評価することができる。

【0 0 7 4】

以上説明した稼働時間、並列処理である処理 1 における $\tau_i(p)$ の和、処理 1 における $\tau_i(p)$ と並列効率の積、非並列処理である処理 2（プロセッサ 4 のみ）における $\tau_i(p)$ の和、及び処理 2 における $\tau_i(p)$ と並列効率の積を図 1 5 に示す。図 1 5 では、並列処理の場合には、稼働時間 T_i より、 $\tau_i(p)$ の方が短く、さらに並列効率を考慮すると無駄な処理時間が除かれるためさらに短くなる。一方、非並列処理の場合には並列効率が 1 となるため、処理 2 における $\tau_i(p)$ も $\tau_i(p)$ と並列効率の積についても同じ値になる。

【0 0 7 5】

並列計算機システムのプロセッサ増設の根拠データとして、従来からシステムの稼働率が用いられてきた。しかし有効に使われたシステムの資源を基にしているわけでないので、並列効率の低い処理のために資源の増設あるいは入れ替えを行う可能性がある。本発明によれば、並列計算機システムのプロセッサ増設に対して定量的指針を与えることが可能になる。システムの全プロセッサ数を P_{System} 、 T_i を各プロセッサの稼働時間、 P_{Add} を増強後のプロセッサ個数、 k_{max} を全処理数、 α を予想並列効率とすると、例えば追加したプロセッサによる以下で

表される稼働時間（数 2 8）だけ増加させたときの加速率 A_{system} は、式（1 8）に示すとおりになる。

【数 2 8】

$$\sum_{i=p_{\text{System}}+1}^{P_{\text{Add}}} T_i$$

【数 2 9】

$$A_{\text{system}} \equiv \frac{\sum_{k=1}^{K_{\max}} \left[E_p(p) \cdot \sum_{i=1}^p \tau_i(p) \right]_k + \alpha \cdot \sum_{i=p_{\text{System}}+1}^{P_{\text{Add}}} T_i}{\sum_{i=1}^{P_{\text{System}}} T_i} \quad (18)$$

【0 0 7 6】

例えば以下に示すような条件で、 $\alpha = 1$ であるとする、加速率 A_{system} は以下のように計算される。

【数 3 0】

$$\sum_{i=1}^{P_{\text{System}}} T_i = 40, \quad \sum_{i=p_{\text{System}}+1}^{P_{\text{Add}}} T_i = 10, \quad \sum_{k=1}^{K_{\max}} \left[E_p(p) \cdot \sum_{i=1}^p \tau_i(p) \right]_k = 39$$

$$A_{\text{system}} = (39 + 1 \times 10) / 40 = 1.23$$

このように約 23% のシステム増強となる。この値は従来の処理の並列効率を考慮しているという点で、増設に対して従来の稼働率より説得力がある値となる。システム増強に予測並列効率 α (< 1) をかけて加速率を算出すれば、より現実的な値となる。また、増強したプロセッサの CPU 能力を 10 倍とするならば、 $\alpha = 10$ として A_{system} を求めることもできる。上記の例に当てはめると、 $A_{\text{system}} = (39 + 10 \times 10) / 40 = 3.48$ となる。これにより異なる CPU 性能のプロセッサの増設に対しても稼働率を基にした以上に根拠がある予測データを作成することが可能となる。

【 0 0 7 7 】

また、本発明により、並列計算機システムの入れ替えに対して定量的指針を与えることも可能となる。各処理について計算された指標など（並列効率、ロードバランス寄与率、仮想並列化率、並列性能阻害要因寄与率、 $\tau_i(p)$ 、 $\gamma_i(p)$ 、 $\chi_{i,j}(p)$ 、各プロセッサの稼動時間 T_i ）により、各処理に対して次の例で示すような並列効率の推測が可能となり、システム入れ替え後のシステムの性能推定が可能となる。

【 0 0 7 8 】

例えば図 1 6 に示したような経過時間が測定されるシステムの CPU 性能を 5 倍にしたシステムを導入することを考えると $\gamma_i(p)$ 、 $\chi_{i,RED}(p)$ 、 $\chi_{i,Other}(p)$ は $1/5$ になる。一方 $\chi_{i,C}(p)$ は、ネットワーク性能に依存し今回は性能が同じとすると新システムの並列効率を次のように推定できる。

【 0 0 7 9 】

なお、 $\chi_{i,C}(1) = 0$ 、 $\chi_{i,Others}(1) = 0$ とすると、CPU 性能が 5 倍となった場合の性能評価指標は次のように計算できる。また、上で述べた式 (12-1) により、 $\chi_{1,RED}(1)$ は以下のように表される。また、ロードバランス寄与率は式 (5) により、仮想並列率は式 (6-1) により、並列性能阻害要因寄与率（冗長処理、通信、その他）は式 (7) に従って、並列効率は式 (4-4) 及び (9-1) に従って、以下のように計算される。

【数 3 1】

$$\chi_{1,RED}(1) = \frac{1}{p} \cdot \sum_{i=1}^p \chi_{i,RED}(p) = \frac{1}{4} \cdot (8+9+7+7)/5 = 1.550$$

$$\begin{aligned} R_b(p) &= \frac{\sum_{i=1}^p \tau_i(p)}{\tau(p) \cdot p} \\ &= \frac{(15+8+1)/5 + 10 + (14+9+1)/5 + 11 + (13+7+1)/5 + 12 + (16+7+1)/5 + 13}{((16+7+1)/5 + 13) \cdot 4} \\ &= \frac{14.8 + 15.8 + 16.2 + 17.8}{17.8 \cdot 4} \\ &= 0.9073 \end{aligned}$$

【数 3 2】

$$R_p(p) = \frac{\sum_{i=1}^p \gamma_i(p)}{\tau(1)} = \frac{(15+14+13+16)/5}{(15+14+13+16)/5 + 7.75/5} = \frac{58/5}{(58+7.75)/5} = 0.8821$$

【数 3 3】

$$R_{RED}(p) = \frac{\sum_{i=1}^4 \chi_{i,RED}(p)}{\sum_{i=1}^4 \tau_i(p)} = \frac{(8+9+7+7)/5}{64.6} = 0.0960$$

【数 3 4】

$$R_C(p) = \frac{\sum_{i=1}^p \chi_{i,C}(p)}{\sum_{i=1}^p \tau_i(p)} = \frac{10+11+12+13}{64.6} = 0.7121$$

【数 3 5】

$$R_{Others}(p) = \frac{\sum_{i=1}^p \chi_{i,others}(p)}{\sum_{i=1}^p \tau_i(p)} = \frac{(1+1+1+1)/5}{64.6} = 0.0124$$

【数 3 6】

$$\begin{aligned} E_p(p) &= R_b(p) \cdot \frac{1}{R_p(p)} \cdot (1 - R_{RED}(p) - R_C(p) - R_{Others}(p)) \\ &= 0.9073 \cdot \frac{1}{0.8821} \cdot (1 - 0.0960 - 0.7121 - 0.0124) = 0.1846 \end{aligned}$$

【数 37】

$$E_p(p) = R_b(p) \cdot \frac{1}{R_p(p)} \cdot \frac{\sum_{i=1}^p \gamma_i(p)}{\sum_{i=1}^p \tau_i(p)} = 0.9073 \cdot \frac{1}{0.8821} \cdot \frac{58/5}{64.6} = 0.1847$$

【0080】

上に示した計算結果と実測に基づく性能指標をまとめると図17に示すようになる。図17の表に示すように、システムを入れ替えた時の並列性能を予測することによって、新しいシステムのシステム運用効率 E_{system} を推定することができる。そのために、今までのシステムのログデータを用い、今までの処理に対して図17と同様にすべての性能指標を計算する。

【0081】

CPU性能を5倍にしたときのシステム運用効率 E_{system} を求める試算をする、以下に示すようになる。推定値を基にして求めたこの E_{system} と今までの実測値から求めた E_{system} を比べることにより、システムの入れ替えに対し、稼働率に比べてより根拠があるデータを得ることができる。

図18に示すように、CPU性能が5倍になった場合の各処理の $E_p(4)$ 、 $\tau_i(p)$ の i についての和、及びプロセッサ数に従って E_{system} が計算できる。なお、前提として以下の条件を用いた。

【数 38】

$$p_{\text{System}} = 10, \quad \sum_{i=1}^{p_{\text{System}}} T_i = 4000, \quad k_{\text{max}} = 4$$

【数 3 9】

$$E_{System} \equiv \frac{\sum_{k=1}^{K_{\max}} \left[E_p(p) \cdot \sum_{i=1}^p \tau_i(p) \right]_k}{\sum_{i=1}^{P_{System}} T_i}$$

$$= \frac{0.1846 \times 64.6 + 0.7219 \times 2000.3 + 0.3000 \times 512.1 + 1 \times 1000}{4000}$$

$$= 0.6524$$

【0 0 8 2】

〔実施の形態の説明〕

図 1 9 に本発明の一実施の形態に係るシステム概要図を示す。並列性能分析装置 1 0 0 は、並列計算機システム 2 0 0 の並列性能を分析する単一プロセッサのコンピュータであり、印刷装置や表示装置といった出力装置 1 1 0 と接続されている。但し、並列性能分析装置 1 0 0 は、並列計算機であってもよい。並列性能分析装置 1 0 0 は、データ取得部 1 0 と、ロードバランス寄与率計算部 1 1 と、仮想並列化率計算部 1 2 と、並列性能阻害要因寄与率計算部 1 3 と、並列効率計算部 1 4 と、補助指標計算部 1 5 と、プロセッサ数最適化処理部 2 1 と、プロセッサ増設見積処理部 2 2 と、システムリプレイスデータ処理部 2 3 と、運用効率データ処理部 2 4 と、チューニング処理部 2 5 と、アルゴリズム選定処理部 2 6 と、並列性能評価処理部 2 7 とを含む。並列性能分析装置 1 0 0 は、ログデータ格納部 3 0 に接続されている。並列計算機システム 2 0 0 は、測定部 2 0 1 を含む。例えば並列性能分析装置 1 0 0 は、並列計算機システム 2 0 0 とネットワークにて接続されている。

【0 0 8 3】

並列計算機システム 2 0 0 の測定部 2 0 1 は、プログラムに従って並列処理を実行しながら、各処理時間 $\gamma_i(p)$ 、 $\chi_{i,j}(p)$ 、 $\tau_i(p)$ を測定する。例えば、各処理の開始から終了までをタイマで計測したり、各処理の開始時刻及び終了時刻を記録して処理終了後に処理時間を計算する。時間の計測は、オペレーティング・

システム（OS：Operating System）を含むソフトウェアによる場合もあれば、ハードウェアによる場合もある。測定された処理時間のデータについては、一旦並列計算機システム 2 0 0 のメモリ中に格納され、場合によっては他の記憶装置に格納される。

【0 0 8 4】

また、処理時間の測定ではなく、一定時間間隔毎に実行中のプログラムの事象を確認し、各事象についてカウントを行う場合もある。このような測定を、サンプリングによる測定と呼ぶ。このようなサンプリングによる測定は、式（4-4）、（9-1）、（9-2）や $Rb(p)$ 、 $Rp(p)$ 、 $Rj(p)$ が時間比の形をしているため採用可能となる。測定精度による違いはあるが、時間測定による方法とサンプリングによる方法では結果は同じになる。

【0 0 8 5】

図 2 0 にサンプリングによる測定の概念図を示す。図 2 0 では左から右に時間が経過する様子を示している。図 2 0 において下向き矢印はサンプリングのタイミングを示しており、下向き矢印の間隔で表されるようにサンプリングは一定時間間隔で行われる。図 2 0 においては、最初に冗長処理が $\chi_{i,RED}(p)$ だけ実施された後、並列計算が $\gamma_i(p)$ だけ行われる。なお、全体として処理は $\tau_i(p)$ だけ実施されている。サンプリング回数は、 $\chi_{i,RED}(p)$ だけ続いた冗長処理の事象においては 7 回、 $\gamma_i(p)$ だけ続いた並列計算の事象においては 9 回である。全体の処理時間 $\tau_i(p)$ の間では、サンプリング回数は 2 2 回である。並列性能阻害要因のうち意図して測定した $\chi_{i,RED}(p)$ 以外の事象をまとめて $\chi_{i,others}(p)$ で表し、意図して測定した $\tau_i(p)$ 、 $\chi_{i,RED}(p)$ 及び $\gamma_i(p)$ を用いて式（13-6）から計算する。図 2 0 の例では、 $\chi_{i,others}(p)$ の間のサンプリング回数が 6 回（ $= 22 - 9 - 7$ ）であることが分かる。

【0 0 8 6】

実際にどのようにサンプリングによる測定を実施するかについては、その概要を以下に説明しておく。

（1） $\tau_i(p)$ の部分

（a）処理の始めにおいて事象 $\tau_i(p)$ のためのフラグを on にし、処理の終了にお

いてoffにする。実行時に事象 $\tau_i(p)$ のためのフラグのon/offを一定時間間隔で識別し、onと識別された回数をカウントしてサンプリング回数を得るものとする。

- ・ 以下の方法のいずれかの記述と処理を、必要に応じて組み合わせて測定する。
- ・ プログラマが、プログラム中処理の始め及び終わり、すなわち上記フラグをon/offすべき位置を検出し、当該フラグをon/offさせるための記述を行う。
- ・ 並列言語拡張やコンパイラ・ディレクティブ等が用いられている場合には、ツールが当該並列言語拡張やコンパイラ・ディレクティブ等を解釈して、上記フラグをon/offさせるための記述を行う。
- ・ 並列言語拡張やコンパイラ・ディレクティブ等が用いられている場合には、コンパイラが当該並列言語拡張やコンパイラ・ディレクティブ等を解釈して、上記フラグをon/offさせるための記述を行う。
- ・ コンパイラが、プログラム中処理の始め及び終わり、すなわち上記フラグをon/offすべき位置を検出し、当該フラグをon/offさせるための記述を行う。
- ・ OSが、プログラム中処理の始め及び終わり、すなわち上記フラグをon/offすべき位置を検出し、当該フラグをon/offさせるための記述を行う。
- ・ ランタイム・ライブラリが、プログラム中処理の始め及び終わり、すなわち上記フラグをon/offすべき位置を検出し、当該フラグをon/offさせるための記述を行う。
- ・ ハードウェアが、プログラム中処理の始め及び終わり、すなわち上記フラグをon/offすべき位置を検出し、当該フラグをon/offさせるための記述を行う。
- ・ 上記フラグがonであることを識別してその回数をカウントする処理のための記述を、コンパイラレベルで行う。
- ・ 上記フラグがonであることを識別してその回数をカウントする処理のための記述を、OSレベルで行う。
- ・ 上記フラグがonであることを識別してその回数をカウントする処理のための記述を、ランタイムライブラリ・レベルで行う。
- ・ 上記フラグがonであることを識別してその回数をカウントする処理のための記述を、ハードウェア・レベルで行う。

- ・ 上記フラグがonであることを識別してその回数をカウントする処理のための記述を、ツールレベルで行う。
- ・ 上記フラグがonであることを識別してその回数をカウントする処理のための記述を、プログラム・レベルで行う。
- ・ 上記フラグがonであることを識別してその回数をカウントする処理の実施を、ハードウェア・レベルで行う。

【 0 0 8 7 】

(b) プログラム名又はそれに代替する実行モジュール名等により事象を特定し、実行時にそのプログラム名又は実行モジュール名等を一定時間間隔で識別し、識別された名称の識別回数をカウントしてサンプリング回数を得るものとする。

以下の方法のいずれかの名前生成法と、識別処理及びカウント処理とを必要に応じて組み合わせて測定する。

- ・ コンパイラが、上記プログラム名又は実行モジュール名等を生成する。
- ・ OS が、上記プログラム名又は実行モジュール名等を生成する。
- ・ ランタイム・ライブラリが、上記プログラム名又は実行モジュール名等を生成する。
- ・ ハードウェアが、上記プログラム名又は実行モジュール名等を生成する。
- ・ 並列言語拡張やコンパイラ・ディレクティブ等の記述により、上記プログラム名又は実行モジュール名等を生成する。
- ・ プログラムの記述により、上記プログラム名又は実行モジュール名等を生成する。
- ・ 生成されたプログラム名又は実行モジュール名等の識別処理及びカウント処理のための記述を、コンパイラ・レベルで行う。
- ・ 生成されたプログラム名又は実行モジュール名等の識別処理及びカウント処理のための記述を、OS レベルで行う。
- ・ 生成されたプログラム名又は実行モジュール名等の識別処理及びカウント処理のための記述を、ランタイムライブラリ・レベルで行う。
- ・ 生成されたプログラム名又は実行モジュール名等の識別処理及びカウント処理のための記述を、ハードウェア・レベルで行う。

- ・生成されたプログラム名又は実行モジュール名等の識別処理及びカウント処理のための記述を、ツール・レベルで行う。
- ・生成されたプログラム名又は実行モジュール名等の識別処理及びカウント処理のための記述を、プログラム・レベルで行う。
- ・生成されたプログラム名又は実行モジュール名等の識別処理及びカウント処理の実施を、ハードウェア・レベルで行う。

【 0 0 8 8 】

(2) $x_{i,j}(p)$ と $r_i(p)$ の部分

(a) 事象 $x_{i,j}(p)$ 、 $r_i(p)$ が出現する毎にその処理の初めにそのためのフラグを on にし、その処理の終わりにそのためのフラグを off にセットする。

【 0 0 8 9 】

実行時に各事象のためのフラグの on/off を一定時間間隔で識別し、on と識別された回数をカウントしてサンプリング回数を得るものとする。1 つの方法では検出できない場合があるため、以下の方法のいずれかの記述と処理を必要に応じて組み合わせて測定する。

- ・プログラマが、プログラム中処理の始め及び終わり、すなわち上記フラグを on/off すべき位置を検出し、当該フラグを on/off させるための記述を行う。
- ・並列言語拡張やコンパイラ・ディレクティブ等が用いられている場合には、ツールが当該並列言語拡張やコンパイラ・ディレクティブ等を解釈して、上記フラグを on/off させるための記述を行う。
- ・並列言語拡張やコンパイラ・ディレクティブ等が用いられている場合には、コンパイラが当該並列言語拡張やコンパイラ・ディレクティブ等を解釈して、上記フラグを on/off させるための記述を行う。
- ・コンパイラが、プログラム中処理の始め及び終わり、すなわち上記フラグを on/off すべき位置を検出し、当該フラグを on/off させるための記述を行う。
- ・OS が、プログラム中処理の始め及び終わり、すなわち上記フラグを on/off すべき位置を検出し、当該フラグを on/off させるための記述を行う。
- ・ランタイム・ライブラリが、プログラム中処理の始め及び終わり、すなわち上記フラグを on/off すべき位置を検出し、当該フラグを on/off させるための記述を

行う。

- ・ハードウェアが、プログラム中処理の始め及び終わり、すなわち上記フラグを on/offすべき位置を検出し、当該フラグを on/offさせるための記述を行う。
- ・上記フラグが onであることを識別してその回数をカウントする処理のための記述を、コンパイラレベルで行う。
- ・上記フラグが onであることを識別してその回数をカウントする処理のための記述を、OSレベルで行う。
- ・上記フラグが onであることを識別してその回数をカウントする処理のための記述を、ランタイムライブラリ・レベルで行う。
- ・上記フラグが onであることを識別してその回数をカウントする処理のための記述を、ハードウェア・レベルで行う。
- ・上記フラグが onであることを識別してその回数をカウントする処理のための記述を、ツールレベルで行う。
- ・上記フラグが onであることを識別してその回数をカウントする処理のための記述を、アプリケーションプログラム・レベルで行う。
- ・上記フラグが onであることを識別してその回数をカウントする処理の実施を、ハードウェア・レベルで行う。

【 0 0 9 0 】

(b) 既知のモジュール名を並列処理部又は並列性能阻害要因に係る処理部に予め分類しておき、実行時にモジュール名を識別し、各モジュール名につきカウントしてサンプリング回数を得る。以下に示す分類方法と、識別処理及びカウント処理を必要に応じて組み合わせて測定する。

- ・モジュール名の分類を、コンパイラレベルで行う。
- ・モジュール名の分類を、OSレベルで行う。
- ・モジュール名の分類を、ランタイムライブラリ・レベルで行う。
- ・モジュール名の分類を、ハードウェア・レベルで行う。
- ・モジュール名の分類を、並列言語拡張やコンパイラディレクティブ・レベルで行う。
- ・モジュール名の分類を、ユーザレベルで行う。

- ・ 上記モジュール名の識別処理及びカウント処理のための記述を、コンパイラ・レベルで行う。
- ・ 上記モジュール名の識別処理及びカウント処理のための記述を、OSレベルで行う。
- ・ 上記モジュール名の識別処理及びカウント処理のための記述を、ランタイムライブラリ・レベルで行う。
- ・ 上記モジュール名の識別処理及びカウント処理のための記述を、ハードウェア・レベルで行う。
- ・ 上記モジュール名の識別処理及びカウント処理のための記述を、ツール・レベルで行う。
- ・ 上記モジュール名の識別処理及びカウント処理のための記述を、プログラム・レベルで行う。
- ・ 上記モジュール名の識別処理及びカウント処理の実施をハードウェア・レベルで行う。

【 0 0 9 1 】

例として、全プロセッサにあるF(I_{max})の各要素を足し合わせる処理をFortranと並列ライブラリのMPI (Message Passing Interface) で記述したプログラムを用いて、 $x_{i,j}(p)$ のサンプリング方法を以下の表1に示す。例えばonのフラグを、コンパイラに指示を行う「*sampon」で表わし、offのフラグを「*sampoﬀ」で表す。また、REDを冗長処理とし、Cを通信とし、REDやCの後ろの数字をその出現順番を表すものとする。総和処理の部分は各プロセッサで同じ計算を行う冗長処理であるので、その始点と終点に「*sampon (RED), 2」及び「*sampoﬀ (RED), 2」を配置し、フラグをon/offする。nLOCALという変数の計算についても冗長処理であるから、その始点と終点に「*sampon (RED), 1」及び「*sampoﬀ (RED), 1」を配置し、フラグをon/offする。さらに、MPI_ALLTOALLは通信ライブラリであり、ここでは「*sampon (C), 1」「*sampoﬀ (C), 1」並びに「*sampon (C), 2」及び「*sampoﬀ (C), 2」を配置し、フラグをon/offする。なお、MPI_ALLTOALLの場合には、ツールやコンパイラやOSが事象を識別してフラグを立てるようにすることも可能である。

【 0 0 9 2 】

【表 1】

```

subroutine GSUM(Imax,F,FW,NP)
  real*8 F(Imax),FW(Imax)
  include 'mpif.h'

*sampon (RED) ,1
  nLOCAL=(Imax+NP-1)/NP
*sampoff (RED) ,1

*sampon (C) ,1
  call MPI_ALLTOALL(F ,nLOCAL,MPI_DOUBLE_PRECISION,
    &                  FW,nLOCAL,MPI_DOUBLE_PRECISION,
    &                  MPI_COMM_WORLD,IERR)
*sampoff (C) ,1

*sampon (RED) ,2
  do j=2,NP
    k=(j-1)*nLOCAL
    do i=1,nLOCAL
      FW(i)=FW(i)+FW(i+k)
    end do
  end do

  do j=2,NP
    k=(j-1)*nLOCAL
    do i=1,nLOCAL
      FW(i+k)=FW(i)
    end do
  end do

```

```

end do
*sampoff (RED) ,2

*sampon (C) ,2
  call MPI_ALLTOALL(FW, nLOCAL,MPI_DOUBLE_PRECISION,
    &                  F ,nLOCAL,MPI_DOUBLE_PRECISION,
    &                  MPI_COMM_WORLD,IERR)
*sampoff (C) ,2
  return
end
【 0 0 9 3 】

```

表 1 のプログラムを実行した際のサンプリングの一例を図 2 1 に示す。図 2 1 では、事象 [(RED),1]、[(RED),2]、[(C),1]、[(C),2] の各々についてカウントされる。但し、並列効率等を計算する際には、冗長処理全体 [(RED),1+(RED),2]、通信処理全体 [(C),1+(C),2] として扱う。

【 0 0 9 4 】

図 1 9 の説明に戻って、並列性能分析装置 1 0 0 のデータ取得部 1 0 は、上で述べたように処理時間又はサンプリング数として測定部 2 0 1 により測定される各処理時間 $\gamma_i(p)$ 、 $\chi_{i,j}(p)$ 、 $\tau_i(p)$ を、並列計算機システム 2 0 0 から取得し、並列性能分析装置 1 0 0 に接続されたログデータ格納部 3 0 に格納する。ログデータ格納部 3 0 には、各処理時間のほかに計算された並列効率を含む並列性能評価指標等のデータも蓄積される。ロードバランス寄与率計算部 1 1 は、上で説明したように式 (5) に従ってロードバランス寄与率 $R_b(p)$ を計算し、記憶装置に格納する。なお、 $\tau(p)$ については式 (2) に従って計算する。仮想並列化率計算部 1 2 は、式 (6-1) に従って仮想並列化率 $R_p(p)$ を計算し、記憶装置に格納する。なお、式 (6-1) の分母の $\tau(1)$ については、式 (8-1) に示したような近似が行われる場合もある。また、式 (10) 及び式 (11) から式 (15) が用いられる場合もある。なお、式 (15) 中の第 2 項 ($\chi_{1,j}(1)$, $j = 1$) については、式 (12-1)、(12-2)、(12-3) 又は (12-4

) のいずれかで計算される場合もある。また、 $x_{1,j}(1)$, $j > 1$ については式 (13-3) 乃至 (13-5) のいずれかで計算される場合もある。 $x_{1,j}(p)$ の $j = 1$ は冗長処理である。但し、他の並列性能阻害要因についての処理時間であってもよい。

【0095】

並列性能阻害要因寄与率計算部 13 は、式 (7) に従って、各並列性能阻害要因につき並列性能阻害要因寄与率 $R_j(p)$ を計算し、記憶装置に格納する。並列効率計算部 14 は、式 (4-4)、式 (4-5)、式 (8-1) の条件が満たされる場合には式 (8-2)、式 (9-1) 又は式 (9-2) のいずれかに従って、並列効率 $E_p(p)$ を計算し、記憶装置に格納する。式 (9-2) を用いる場合に分子の第 1 項については式 (12-1)、(12-2)、(12-3)、(12-4)、(13-3)、(13-4) 又は (13-5) のいずれかで計算される場合もある。式 (12-1) 乃至 (12-4) は、 $x_{1,j}(p)$, $j = 1$ の冗長処理である。補助指標計算部 15 は、例えば式 (6-2) に従って加速率 A_p を、並列効率 $E_p(p)$ とプロセッサ数 p とから $E_p(p) \cdot p$ を計算し、記憶装置に格納する。

【0096】

プロセッサ数最適化処理部 21 は、並列計算機システム 200 のエンドユーザに、処理のために投入すべき最適なプロセッサ数を指示するための処理等を実施する。プロセッサ増設見積処理部 22 は、並列計算機システム 200 の運用管理者に、プロセッサの増設に際して指針となる数値を提示するための処理を実施する。システムリプレイスデータ処理部 23 は、並列計算機システム 200 の運用管理者に、システムリプレイスに際して指針となる数値を提示するための処理を実施する。運用効率データ処理部 24 は、並列計算機システム 200 の運用管理者に、システム運用効率に関するデータを提示するための処理を実施する。チューニング処理部 25 は、並列処理を行うためのプログラムに、プログラマが適切な目標設定等により効率的なプログラム等のチューニングを実施できるようにするための処理を実施する。アルゴリズム選定処理部 26 は、同一処理について異なるアルゴリズムが存在する場合に、並列処理を行うためのプログラムに、プログラマが並列効率をより向上させることのできるアルゴリズムを選択できるように

にするための処理等を実施する。並列性能評価処理部 2 7 は、並列計算機システムの開発者や研究者が並列性能の評価を容易に行うことができるようにするための処理を実施する。これらの処理部の詳細な処理内容については、以下で説明する。

【 0 0 9 7 】

次に図 1 9 に示したシステム等の処理フローを図 2 2 を用いて説明する。最初に、処理時間の直接の計測のための記述、コンパイラ、OS、ツール、プログラマ、ランタイム・ライブラリ、ハードウェア等により各処理時間に対応するサンプリング数をカウントするためのフラグを on/off させるための記述、コンパイラ、OS、ツール、プログラマ、ランタイム・ライブラリ、ハードウェア等により各処理時間に対応するサンプリング数をカウントするためのモジュール名等の分類などを含む前処理を実施する（ステップ S 1）。この処理については、並列計算機システム 2 0 0 で行われる場合もあれば、他の計算機システムにおいて行われる場合もある。さらに、プログラマなどの人間により行われる場合もある。なお、ステップ S 1 は、並列性能分析装置 1 0 0 において実施される処理ではなく並列計算機システム 2 0 0 により実施される処理でもない場合もあるので、点線ブロックで表されている。

【 0 0 9 8 】

次に、並列計算機システム 2 0 0 の測定部 2 0 1 は、前処理に基づいて、処理時間の計測を実施したり、サンプリング数をカウントしたりする測定処理を実施する（ステップ S 3）。測定結果である各処理時間 $\gamma_i(p)$ 、 $x_{i,j}(p)$ 、 $\tau_i(p)$ 又は各処理時間に対応するサンプリングのカウント値については、並列計算機システム 2 0 0 の記憶装置に格納され、並列性能分析装置 1 0 0 のデータ取得部 1 0 により読み出される。データ取得部 1 0 は、各処理時間 $\gamma_i(p)$ 、 $x_{i,j}(p)$ 、 $\tau_i(p)$ 又は各処理時間に対応するサンプリングのカウント値を取得すると、並列性能分析装置 1 0 0 のログデータ格納部 3 0 に格納する。

【 0 0 9 9 】

そして、ロードバランス寄与率計算部 1 1、仮想並列化率計算部 1 2、並列性能阻害要因寄与率計算部 1 3、並列効率計算部 1 4、補助指標計算部 1 5 は、ロ

グデータ格納部 3 0 に格納された各処理時間 $\gamma_i(p)$ 、 $\chi_{i,j}(p)$ 、 $\tau_i(p)$ 又は各処理時間に対応するサンプリングのカウント値等を用いて、ロードバランス寄与率 $Rb(p)$ 、仮想並列化率 $Rp(p)$ 、各並列性能阻害要因寄与率 $Rj(p)$ 、並列効率 $E_p(p)$ 、加速率 A_p その他の補助指標を計算し、ログデータ格納部 3 0 に格納する（ステップ S 5）。

【 0 1 0 0 】

並列効率 $E_p(p)$ については、上で述べたように式（4-4）、式（4-5）、式（8-1）の条件が満たされる場合には式（8-2）、式（9-1）又は式（9-2）のいずれかに従って計算される。従って、並列効率計算部 1 4 は、ロードバランス寄与率計算部 1 1 により計算されたロードバランス寄与率 $Rb(p)$ 、仮想並列化率計算部 1 2 により計算された仮想並列化率 $Rp(p)$ 、並列性能阻害要因寄与率計算部 1 3 により計算された各並列性能阻害要因寄与率 $Rj(p)$ 、補助指標計算部 1 5 により計算された加速度 $A_p(p)$ を用いて、その他の必要なデータについてはログデータ格納部 3 0 に格納された処理時間等を用いて、並列効率 $E_p(p)$ を計算する。

【 0 1 0 1 】

例えば図 2 3 のような処理時間の測定結果がデータ取得部 1 0 によりログデータ格納部 3 0 に格納された場合の計算例を以下に示す。より具体的には、 $\tau_1(p) = 34$ 、 $\tau_2(p) = 35$ 、 $\tau_3(p) = 33$ 、 $\tau_4(p) = 37$ 、 $\gamma_1(p) = 15$ 、 $\gamma_2(p) = 14$ 、 $\gamma_3(p) = 13$ 、 $\gamma_4(p) = 16$ 、 $\chi_{1,RED}(p) = 8$ 、 $\chi_{2,RED}(p) = 9$ 、 $\chi_{3,RED}(p) = 7$ 、 $\chi_{4,RED}(p) = 7$ 、 $\chi_{1,C}(p) = 10$ 、 $\chi_{2,C}(p) = 11$ 、 $\chi_{3,C}(p) = 12$ 、 $\chi_{4,C}(p) = 13$ という測定結果が得られたものとする。従って、 $\chi_{1,others}(p) = 1$ ($= 34 - 15 - 8 - 10$)、 $\chi_{2,others}(p) = 1$ ($= 35 - 14 - 9 - 11$)、 $\chi_{3,others}(p) = 1$ ($= 33 - 13 - 7 - 12$)、 $\chi_{4,others}(p) = 1$ ($= 37 - 16 - 7 - 13$) となる。以下の計算で必要となるが未測定のもの $\chi_{1,C}(1)$ 及び $\chi_{1,others}(1)$ については共に 0 とする。

【 0 1 0 2 】

(1) ロードバランス寄与率（式（5））

【数 4 0】

$$R_b(p) = \frac{\sum_{i=1}^p \tau_i(p)}{\tau(p) \cdot p} = \frac{34 + 35 + 33 + 37}{37 \cdot 4} = \frac{139}{148} = 0.9392$$

(2) 仮想並列化率 (式 (6-1), (12-1), (15))

【数 4 1】

$$\begin{aligned} \chi_{1,RED}(1) &\equiv \frac{1}{p} \cdot \sum_{i=1}^p \chi_{i,RED}(p) = \frac{1}{4} \cdot (8 + 9 + 7 + 7) = 7.75 \\ \tau_1(1) &= \sum_{i=1}^p \gamma_i(p) + \sum_{j=1}^{j_{others}} \chi_{1,j}(1) = (15 + 14 + 13 + 16) + 7.75 = 65.75 \\ R_p(p) &= \frac{\sum_{i=1}^p \gamma_i(p)}{\tau(1)} = \frac{15 + 14 + 13 + 16}{65.75} = \frac{58}{65.75} = 0.8821 \end{aligned}$$

(3) 加速率 (式 (6-2))

【数 4 2】

$$A_p(p) = \frac{1}{1 - R_p(p)} = \frac{1}{1 - 0.8821} = 8.482$$

(4) 並列性能阻害要因寄与率 (式 (7))

(4-1) 冗長処理の並列性能阻害要因寄与率

【数 4 3】

$$R_{RED}(p) = \frac{\sum_{i=1}^4 \chi_{i,RED}(p)}{\sum_{i=1}^4 \tau_i(p)} = \frac{(8 + 9 + 7 + 7)}{139} = 0.2230$$

(4-2) 通信処理の並列性能阻害要因寄与率

【数 4 4】

$$R_C(p) = \frac{\sum_{i=1}^p \chi_{i,C}(p)}{\sum_{i=1}^p \tau_i(p)} = \frac{10+11+12+13}{139} = 0.3309$$

(4-3) その他の並列性能阻害要因寄与率

【数 4 5】

$$R_{Others}(p) = \frac{\sum_{i=1}^p \chi_{i,others}(p)}{\sum_{i=1}^p \tau_i(p)} = \frac{1+1+1+1}{139} = 0.0288$$

【 0 1 0 3】

(5-1) 並列効率 (式 (4-4))

【数 4 6】

$$\begin{aligned} E_p(p) &= R_b(p) \cdot \frac{1}{R_p(p)} \cdot (1 - R_{RED}(p) - R_C(p) - R_{Others}(p)) \\ &= 0.9392 \cdot \frac{1}{0.8821} \cdot (1 - 0.2230 - 0.3309 - 0.0288) = 0.4443 \end{aligned}$$

(5-2) 並列効率 (式 (9-1))

【数 4 7】

$$E_p(4) = R_b(4) \cdot \frac{1}{R_p(4)} \cdot \frac{\sum_{i=1}^p \gamma_i(4)}{\sum_{i=1}^p \tau_i(4)} = 0.9392 \cdot \frac{1}{0.8821} \cdot \frac{58}{139} = 0.4443$$

(5-3) 並列効率 (式 (9-2))

【数 4 8】

$$E_p(p) \equiv \frac{\chi_{1,RED}(1) + \sum_{i=1}^p \gamma_i(p)}{\tau(p) \cdot p} = \frac{7.75 + 58}{37 \cdot 4} = 0.4443$$

【0 1 0 4】

以上の結果をまとめると図 1 3 のようになる。なお、補助指標である $E_p(p) \cdot p$ も計算されている。 $E_p(4) \cdot p = 1.777$ で、4 プロセッサ並列で 1.777 プロセッサの性能で処理していることが分かる。並列効率はロードバランスで約 94% ($= R_b(4) = 0.9392$) に低下する。並列性能阻害要因の影響は、冗長処理が約 22% ($= R_{RED}(4) = 0.2230$)、通信が約 33% ($= R_C(4) = 0.3309$)、その他が 3% ($= R_{others}(4) = 0.0288$) である。従って主に通信と冗長処理実行で 55% 程度並列効率を下げている。また図 1 3 では $R_p(4) = 0.8821$ ということから、プロセッサを無限に投入した時の並列最大性能が 1 プロセッサの 8.482 ($= A_p(4) = 1 / (1 - 0.8821)$) 倍であることが推定できる。従って、この処理は 8 プロセッサ以下で行われるべき処理であることが分かる。この処理を $E_p(x) \geq 0.8$ の条件で行おうとすれば、 $E_p(x) \cdot p = 0.4443 \times 4 = 1.777 = E_p(x) \cdot x = 0.8 \cdot x$ で、 $x = 2.22$ となる。従って $p \geq 2.22 = 2$ が与えられた条件に対する最適なプロセッサ数と予想できる。

【0 1 0 5】

また、例えば図 2 4 のようなサンプリングによるカウント数がデータ取得部 10 によりログデータ格納部 30 に格納された場合の計算例を以下に示す。より具体的には、 $\tau_1(p) = 3488$ 、 $\tau_2(p) = 3561$ 、 $\tau_3(p) = 3372$ 、 $\tau_4(p) = 3756$ 、 $\gamma_1(p) = 1521$ 、 $\gamma_2(p) = 1411$ 、 $\gamma_3(p) = 1322$ 、 $\gamma_4(p) = 1601$ 、 $\chi_{1,RED}(p) = 823$ 、 $\chi_{2,RED}(p) = 945$ 、 $\chi_{3,RED}(p) = 711$ 、 $\chi_{4,RED}(p) = 703$ 、 $\chi_{1,C}(p) = 1056$ 、 $\chi_{2,C}(p) = 1111$ 、 $\chi_{3,C}(p) = 1230$ 、 $\chi_{4,C}(p) = 1341$ という測定結果が得られたものとする。従って、 $\chi_{1,others}(p) = 88$ ($= 3488 - 1521 - 823 - 1056$)、 $\chi_{2,others}(p) = 94$ ($= 3561 - 1411 - 945 - 1111$)、 $\chi_{3,others}(p) = 109$

(= 3372 - 1322 - 711 - 1230)、 $\chi_{4, \text{others}}(p) = 111$ (= 3756 - 1601 - 703 - 1341) となる。以下の計算で必要となるが未測定のもの $\chi_{1, C}(1)$ 及び $\chi_{1, \text{others}}(1)$ については共に0とする。

【0106】

(1) ロードバランス寄与率 (式 (5))

【数49】

$$R_b(4) = \frac{\sum_{i=1}^4 \tau_i(4)}{\tau(4) \cdot 4}$$

$$= \frac{(1521+823+1056+88) + (1411+945+1111+94) + (1322+711+1230+109) + (1601+703+1341+111)}{(1601+703+1341+111) \cdot 4}$$

$$= \frac{3488+3561+3372+3756}{3756 \cdot 4} = \frac{14177}{15024} = 0.9436$$

(2) 仮想並列化率 (式 (6-1), (12-1), (15))

【数50】

$$\chi_{1, RED}(1) = \frac{1}{P} \cdot \sum_{i=1}^P \chi_{i, RED}(p) = \frac{1}{4} \cdot (823+945+711+703) = 795.5$$

$$R_p(p) = \frac{\sum_{i=1}^P \gamma_i(p)}{\tau(1)} = \frac{1521+1411+1322+1601}{(1521+1411+1322+1601) + 795.5} = \frac{5855}{5855+795.5} = 0.8804$$

(3) 加速率 (式 (6-2))

【数51】

$$A_p(p) = \frac{1}{1 - R_p(p)} = \frac{1}{1 - 0.8804} = 8.361$$

(4) 並列性能阻害要因寄与率 (式 (7))

(4-1) 冗長処理の並列性能阻害要因寄与率

【数 5 2】

$$R_{RED}(4) = \frac{\sum_{i=1}^4 \chi_{i,RED}(4)}{\sum_{i=1}^4 \tau_i(4)} = \frac{(823+945+711+703)}{14177} = 0.2244 \quad (\text{冗長処理})$$

(4-2) 通信処理の並列性能阻害要因寄与率

【数 5 3】

$$R_C(p) = \frac{\sum_{i=1}^p \chi_{i,C}(p)}{\sum_{i=1}^p \tau_i(p)} = \frac{1056+1111+1230+1341}{14177} = 0.3342 \quad (\text{通信})$$

(4-3) その他の並列性能阻害要因寄与率

【数 5 4】

$$R_{Others}(p) = \frac{\sum_{i=1}^p \chi_{i,others}(p)}{\sum_{i=1}^p \tau_i(p)} = \frac{88+94+109+111}{14177} = 0.0284 \quad (\text{その他})$$

【0 1 0 7】

(5-1) 並列効率 (式 (4-4))

【数 5 5】

$$\begin{aligned} E_p(p) &= R_b(p) \cdot \frac{1}{R_p(p)} \cdot (1 - R_{RED}(p) - R_C(p) - R_{Others}(p)) \\ &= 0.9436 \cdot \frac{1}{0.8804} \cdot (1 - 0.2244 - 0.3342 - 0.0284) = 0.4426 \end{aligned}$$

(5-2) 並列効率 (式 (9-1))

【数 5 6】

$$E_p(p) = R_b(p) \cdot \frac{1}{R_p(p)} \cdot \frac{\sum_{i=1}^p \gamma_i(p)}{\sum_{i=1}^p \tau_i(p)} = 0.9436 \cdot \frac{1}{0.8804} \cdot \frac{5855}{14177} = 0.4426$$

(5-3) 並列効率 (式 (9-2))

【数 5 7】

$$E_p(p) \equiv \frac{\chi_{1,RED}(1) + \sum_{i=1}^p \gamma_i(p)}{\tau(p) \cdot p} = \frac{795.5 + 5855}{3756 \cdot 4} = 0.4427$$

以上の結果をまとめると処理時間の計測の場合と同じ図 1 3 のようになる。

【0 1 0 8】

図 2 2 の説明に戻って、ログデータ格納部 3 0 には、処理毎に処理時間等の測定結果と図 1 3 に示すような並列性能評価指標及び補助指標とがセットで格納される。そして、並列性能分析装置 1 0 0 は、ユーザによる要求に応じて又は自動的に、表示装置や印刷装置等の出力装置 1 1 0 に、図 1 3 のような処理結果を出力する (ステップ S 7)。

【0 1 0 9】

ユーザは、図 1 3 のようなデータだけで自ら並列性能などについて分析、最適プロセッサ数の見積、プロセッサ増設やシステムリプレイスを行う場合の効果の見積、プログラム等のチューニング、アルゴリズムの選定などを行っても良い。しかし、プロセッサ数最適化処理部 2 1、プロセッサ増設見積処理部 2 2、システムリプレイスデータ処理部 2 3、運用効率データ処理部 2 4、チューニング処理部 2 5、アルゴリズム選定処理部 2 6、並列性能評価処理部 2 7 により、以下で説明するような各種コンサルティング支援処理を、ユーザの指示に従って実施する (ステップ S 9)。これにより、より具体的なデータを得ることができるようになる。

【0 1 1 0】

A. プロセッサ数最適化処理

プロセッサ数最適化処理部 2 1 による処理を図 2 5 及び図 2 6 を用いて説明する。プロセッサ数最適化処理部 2 1 は、ユーザによる目標並列効率 $(E_p)_T$ の値の設定入力を受け付ける（ステップ S 1 1）。そして、最適プロセッサ数の計算を以下の式に従って行い、記憶装置に格納する（ステップ S 1 3）。

$$(p)_{OPT} = E_p(p) / (E_p)_T \cdot p$$

そして、計算された最適プロセッサ数を出力装置 1 1 0 に出力する（ステップ S 1 5）。これにより、ユーザは、次に同じ処理群に属する処理を実施する際に使用するプロセッサを必要最小限にすることができる。例えば、上でも説明しているが、図 1 3 のような計算結果が得られており、目標並列効率 $(E_p)_T = 0.8$ とすると、 $p = 2.22$ なる。従って最適なプロセッサ数は 2 となる。

【0 1 1 1】

また、連続して同じ処理群の処理を実施する場合には、最適プロセッサ数を調整しながらより効率的に処理を実施させることも可能になる。すなわち、図 2 6 に示すような処理を実施する。最初に、プロセッサ数 p の仮設定を行う（ステップ S 2 1）。この仮設定されたプロセッサ数 p は同じ処理群の最初の処理について用いられる。また、ユーザから目標並列効率の設定を受け付ける（ステップ S 2 3）。そして、プロセッサ数 p の設定に従って、並列計算機システム 2 0 0 により並列処理を実施すると共に測定部 2 0 1 により処理時間等を測定し、記憶装置に測定結果を格納する（ステップ S 2 5）。データ取得部 1 0 は、測定部 2 0 1 により測定された処理時間等のデータをログデータ格納部 3 0 に格納する。そして、並列効率計算部 1 4 等により、並列効率を含む並列性能評価指標等を計算し、ログデータ格納部 3 0 に格納する（ステップ S 2 7）。

【0 1 1 2】

そしてプロセッサ数最適化処理部 2 1 は、上で述べた式に従って最適プロセッサ数 $(p)_{OPT}$ を計算し、記憶装置に格納する（ステップ S 2 9）。この計算された最適プロセッサ数 $(p)_{OPT}$ を、同じ処理群の次の処理にて使用するプロセッサ数として p に設定する（ステップ S 3 1）。そして、同一処理群の全ての処理を実施したか判断する（ステップ S 3 3）。もし、全ての処理を実施したわけでは

ない場合には、同一処理群の次の処理を選択し（ステップ S 3 5）、ステップ S 2 5 に戻り、ステップ S 3 1 において設定したプロセッサ数で並列処理を実施する。

【 0 1 1 3 】

このような処理を実施することにより、同一処理群に属する前の処理についての最適プロセッサ数を次の処理のプロセッサ数として設定することができるので、より効率的に処理群の処理を行うことができるようになる。

【 0 1 1 4 】

B. プロセッサ増設見積処理

プロセッサ増設見積処理部 2 2 は、並列計算機システム 2 0 0 のプロセッサ増設に対して定量的な指針として、システム増強時の加速率 A_{system} を与えるための処理を実施する。図 2 7 に処理フローを示す。まず、プロセッサ増設見積処理部 2 2 は、システム増設時の増加分の稼働時間のデータ及びその予想並列効率のデータの設定入力を受け付ける（ステップ S 4 1）。そして、式（1 8）に従ってシステム増設時の加速率 A_{system} を計算し、記憶装置に格納する（ステップ S 4 3）。なお、現在使用中の各プロセッサの稼働時間等のデータについては、ログデータ格納部 3 0 に格納された過去の処理ログデータを用いて計算する。そして、計算されたシステム増設時の加速率 A_{system} を表示装置などの出力装置 1 1 0 に出力する（ステップ S 4 5）。

【 0 1 1 5 】

設定した増加分の稼働時間とその予測並列効率に対するシステム増設時の加速率 A_{system} により、どの程度意味のある処理を実施するための時間が増加するかといったことを判断することができるようになる。

【 0 1 1 6 】

C. システムリプレイスデータ処理

並列計算機システムの入れ替えに際して新しい並列計算機システムの性能決定のための定量的指針を提示するための処理を実施する。図 2 8 にそのための処理フローを示す。システムリプレイスデータ処理部 2 3 は、目標とする並列効率（ E_p ）_T 及び繰返回数 i_{cmax} の設定入力を受け付ける（ステップ S 5 1）。また、

新しい並列計算機システムの性能として現行の並列計算機システムに対する性能倍率 A の設定入力を受け付ける（ステップS53）。性能倍率については、CPU性能の倍率 A_{CPU} 、通信性能の倍率 A_C 、I/O性能の倍率 $A_{I/O}$ 等の設定入力を受け付ける。定量的指針はこの倍率値で得られる。ほとんどの計算機システム入れ替えでは、CPU性能の改善によりシステムの性能向上を図るため、例えばまず A_{CPU} を設定し、他の性能倍率を1として E_p を計算する。そして $(E_p)_T$ に近付くように A_C 、 $A_{I/O}$ 等の値を繰り返し計算により求め、新しい並列計算機システムの性能決定のための指針を得るような方針で処理を行ってもよい。

【0117】

より具体的には、システムリプレイスデータ処理部23は、設定された各性能倍率に従ってログデータ格納部30に格納された各処理時間等を短くする計算を実施する（ステップS55）。例えばCPU性能が5倍（ $A_{CPU}=5$ ）と設定された場合には、並列処理の処理時間 $\gamma_i(p)$ などを5分の1にするといった計算を行う。そして、設定された各性能倍率に従って短縮された各処理時間に基づき、並列効率を含む並列性能評価指標の見積値（例えば並列効率の見積値は $(E_p)_E$ ）を計算し、記憶装置に格納する（ステップS57）。

【0118】

システムリプレイスデータ処理部23は、並列効率の見積値 $(E_p)_E$ が目標の並列効率 $(E_p)_T$ とが一致するか判断する（ステップS59）。完全一致でなくてもよく、目標の並列効率 $(E_p)_T$ の所定の範囲内に見積値 $(E_p)_E$ が入っているか判断する。もし、並列効率の見積値 $(E_p)_E$ が目標の並列効率 $(E_p)_T$ とほぼ一致すると判断された場合には、目標並列効率達成を示すメッセージ及びステップS57において計算された各並列性能評価指標の見積値等を、表示装置などの出力装置110に出力する（ステップS61）。一方、並列効率の見積値 $(E_p)_E$ が目標の並列効率 $(E_p)_T$ とがほぼ一致しているとは言えない場合、カウンタ i_c が繰返回数 i_{cmax} 以上になったか判断する（ステップS63）。もし、カウンタ i_c が繰返回数 i_{cmax} 以上になった場合には、目標並列効率を達成できなかった旨のメッセージ及びステップS57において計算された各並列能力評価指標の見積値等を表示装置等の出力装置110に出力する（ステップS65）。

【0119】

一方、 i_c が繰返回数 i_{cmax} 未満である場合には、CPU性能の倍率、通信性能の倍率、I/O性能の倍率等の性能倍率の変更を実施する（ステップS67）。このステップについては、自動的に変更するようにしてもよいし、ユーザによる設定を受け付けるようにしてもよい。そして、カウンタ i_c を1インクリメントし（ステップS69）、ステップS55に戻る。

【0120】

上の処理では目標の並列効率 $(E_p)_T$ を達成するように、最大繰返回数 i_{cmax} まで性能倍率を変更して並列効率の見積を実施する。なお、ログデータ格納部30に処理時間等が格納されている特定の処理に対して $(E_p)_T$ を満足する性能を有する新たな並列計算機システムを選ぶことも、幾つかの種類の処理に対して $(E_p)_T$ を満足する性能を有する新たな並列計算機システムを選ぶことも可能となる。

【0121】

図28の処理フローの適用例を、具体的に図23のような処理時間が測定されたケースで説明する。ここでは、目標並列効率 $(E_p)_T = 0.6$ とし、CPU性能が5倍、すなわち $A_{CPU} = 5$ である新たなシステムを導入することを想定すると、 $\gamma_i(p)$ 、 $\chi_{i,RED}(p)$ は、 $1/A_{CPU}$ となる。性質が不明な $\chi_{i,others}(p)$ も $1/A_{CPU}$ となるものと仮定する。一方、 $\chi_{i,C}(p)$ はネットワーク性能に依存する。ここでは、まず最初に、 $A_C = \infty$ として実現可能性を検討する。なお、未測定のもの $\chi_{1,C}(1)$ 及び $\chi_{1,others}(1)$ については共に0とする。

【0122】

式(12-1)から以下のような計算がなされる。

【数58】

$$\chi_{1,RED}(1) \equiv \frac{1}{p} \cdot \sum_{i=1}^p \chi_{i,RED}(p) = \frac{1}{4} \cdot (8+9+7+7) = 7.75$$

[$A_{CPU} = 5$, $A_C = \infty$ の場合の $(E_p)_E$]

【数59】

$$\begin{aligned}
 R_b(p) &= \frac{\sum_{i=1}^p \tau_i(p)}{\tau(p) \cdot p} \rightarrow \frac{\sum_{i=1}^p ((\gamma_i(p) + \chi_{i,RED}(p) + \chi_{i,others}(p)) / A_{CPU} + \chi_{i,C}(p) / A_C)}{\tau(p) \cdot p} \\
 &= \frac{(15+8+1)/5+10/\infty + (14+9+1)/5+11/\infty + (13+7+1)/5+12/\infty + (16+7+1)/5+13/\infty}{((16+7+1)/5+13/\infty) \cdot 4} \\
 &= \frac{(24+24+21+24)/5}{24/5 \cdot 4} = \frac{18.6}{19.2} \\
 &= 0.9688
 \end{aligned}$$

【数60】

$$\begin{aligned}
 R_p(p) &= \frac{\sum_{i=1}^p \gamma_i(p)}{\tau(1)} \rightarrow \frac{\sum_{i=1}^p \gamma_i(p) / A_{CPU}}{\sum_{i=1}^p \gamma_i(p) / A_{CPU} + \chi_{1,RED}(1) / A_{CPU}} = \frac{\sum_{i=1}^p \gamma_i(p)}{\sum_{i=1}^p \gamma_i(p) + \chi_{1,RED}(1)} \\
 &= \frac{(15+14+13+16)/5}{(15+14+13+16)/5 + 7.75/5} = \frac{58}{(58+7.75)} = 0.8821
 \end{aligned}$$

【数61】

$$\begin{aligned}
 R_{RED}(p) &= \frac{\sum_{i=1}^4 \chi_{i,RED}(p)}{\sum_{i=1}^4 \tau_i(p)} \rightarrow \frac{\sum_{i=1}^4 \chi_{i,RED}(p) / A_{CPU}}{\sum_{i=1}^4 [(\gamma_i(p) + \chi_{i,RED}(p) + \chi_{i,others}(p)) / A_{CPU} + \chi_{i,C}(p) / A_C]} \\
 &= \frac{(8+9+7+7)/5}{18.6} = 0.3333
 \end{aligned}$$

【数62】

$$\begin{aligned}
 R_{others}(p) &= \frac{\sum_{i=1}^4 \chi_{i,others}(p)}{\sum_{i=1}^4 \tau_i(p)} \rightarrow \frac{\sum_{i=1}^4 \chi_{i,others}(p) / A_{CPU}}{\sum_{i=1}^4 [(\gamma_i(p) + \chi_{i,RED}(p) + \chi_{i,others}(p)) / A_{CPU} + \chi_{i,C}(p) / A_C]} \\
 &= \frac{(1+1+1+1)/5}{18.6} = 0.0430
 \end{aligned}$$

【数 6 3】

$$\begin{aligned}
 E_p(p) &= R_b(p) \cdot \frac{1}{R_p(p)} \cdot (1 - R_{RED}(p) - R_C(p) - R_{Others}(p)) \\
 &= 0.9688 \cdot \frac{1}{0.8821} \cdot (1 - 0.3333 - 0 - 0.0430) = 0.6850
 \end{aligned}$$

【0 1 2 3】

以上の計算結果などを図 2 9 にまとめる。 $A_C = \infty$ では $\chi_{i,C}(p)$ の項は 0 となり、 $E_p = 0.6850$ となるので目標値 0.6 より大きい。従って目標並列効率が A_C の性能向上次第で達成できる可能性があることが分かる。そこでステップ S 6 7 で A_C を変更しながら繰り返し並列効率の計算を実施し（ステップ S 5 7）、 $E_p(p) \sim 0.6$ となる A_C を探す。途中の計算は省略するが、 $E_p(p) \sim 0.6$ の場合の計算結果を図 2 9 の第 2 行に示す。この場合、 $A_C = 19.2$ である。これより $(E_p)_T = 0.6$ で CPU 性能を $A_{CPU} = 5$ としたい場合、 $A_C = 19.2$ 以上の性能を有する並列計算機システムを探して入れ替えればよいという指針が得られる。

【0 1 2 4】

なお、 $A_C = 19.2$ という数字が高すぎて、そのようなシステムが現存しない場合には、他の並列性能阻害要因の縮小を検討する。図 2 9 の第二行の見積結果によれば、冗長処理 $R_{RED}(4) = 0.2953$ を改善すべきということが分かる。冗長処理を削減するためには、プログラムのチューニングが必要となる。チューニングしたプログラムを実行し、図 2 8 の処理を再度実行して、 A_C を再度見積もればよい。

【0 1 2 5】

また、図 2 9 に示したようにシステムリプレイス時の並列性能を予測することで、新たに導入する並列計算機システムのシステム運用効率 E_{system} を推定することも可能である。例えば、ある処理で目標並列効率 $(E_p)_T = 0.6$ をクリアする、 $A_{CPU} = 5$ 、 $A_C = 19.2$ という新しいシステムで、見積りが図 1 8 に示される処理 1 乃至 4 を行った場合、 $E_{system} = 0.6534$ となるということが分かる。この予測された E_{system} と、これまでの処理ログから得られる E_{system} とを比較

することにより、システムの入れ替えによる稼働率の向上をより根拠があるデータで定量的に示すことができるようになる。

【0 1 2 6】

D. システム運用効率向上処理

例えば式(17)で示されたシステム運用効率 E_{system} という指標を基に、システムの運用効率を評価する。この指標向上のために、どの処理の並列効率をどの程度向上させる必要があるか等、運用効率の向上に関して具体的指針を出す。具体的には、図30の処理を実施する。

【0 1 2 7】

運用効率データ処理部24は、運用管理者によるシステム運用効率の目標値($E_{\text{system}})_T$ 及び繰返回数 i_{cmax} の設定入力を受け付ける(ステップS71)。そして、ログデータ格納部30に格納されている処理時間や並列効率等のデータを読み出し、式(17)に従ってシステム運用効率 E_{system} を計算し、記憶装置に格納する(ステップS73)。なお、並列効率を含む並列性能評価指標の計算がなされていない場合には、この段階にてロードバランス寄与率計算部11、仮想並列化率計算部12、並列性能阻害要因寄与率計算部13、並列効率計算部14などにより並列効率を含む並列性能評価指標などを計算する。そして、ステップS73において計算されたシステム運用効率 E_{system} がシステム運用効率の目標値($E_{\text{system}})_T$ を超えたか判断する(ステップS75)。もし、 $E_{\text{system}} > (E_{\text{system}})_T$ であると判断された場合には、目標達成を表すメッセージ及びステップS73において計算されたシステム運用効率 E_{system} を表示装置などの出力装置110に出力する(ステップS77)。一方、 $E_{\text{system}} \leq (E_{\text{system}})_T$ である場合には、カウンタ値 i_c が繰返回数 i_{cmax} 以上になっているか判断する(ステップS79)。もし、カウンタ値 i_c が繰返回数 i_{cmax} 以上であれば、システム運用効率向上処理がうまく機能していないことを知らせるため、目標未達を示すメッセージ及び直前のステップS73で計算されたシステム運用効率 E_{system} を、表示装置などの出力装置110に出力する(ステップS81)。

【0 1 2 8】

一方、カウンタ値 i_c が繰返回数 i_{cmax} 未満であれば、運用効率データ処理部

24 は、エンドユーザにはエンドユーザ向けの改善処置を、システム管理者にはシステム管理者向けの改善処置を、プログラマにはプログラマ向けの改善処置を、並列計算機システム開発者又は研究者には並列計算機システム開発者又は研究者向けの改善処置を実施するように勧め、エンドユーザ等は可能なシステム運用効率改善処置を実施する（ステップ S 8 3）。なお、実施する処置の例としては、プロセッサ数を最適化させたり、プロセッサの増設、システムのリプレイス、プログラム等のチューニング等である。システム運用効率改善処置の実施後に、再度並列計算機システム 2 0 0 により並列処理を実施し、同時に測定部 2 0 1 による処理時間等の測定処理を実施する（ステップ S 8 5）。そして、カウンタ値 ic を 1 インクリメントし（ステップ S 8 7）、ステップ S 7 3 に戻る。なお、ステップ S 8 3 についてはエンドユーザなどが行う処理である場合もあるので点線ブロックで、ステップ S 8 5 については並列性能分析装置 1 0 0 の処理ではないので一点鎖線のブロックで示している。

【 0 1 2 9 】

このような処理を実施することにより、従来の稼働率 NWR_{system} では考慮されていない並列効率を反映させた、すなわち実効的な処理時間を考慮に入れたシステム運用効率を向上させることができるようになる。

【 0 1 3 0 】

E. チューニング処理

従来、並列アプリケーション・プログラムのチューニングによる性能向上作業は、達成目標が不明確であったためその作業時間の見積が容易でなかった。目標とした処理時間がチューニングでは到達不可能な場合もあり、チューニング作業を延々と続けて多大な作業時間を費やす場合も多々存在していた。そこで、図 3 1 に示すような処理を実施する。

【 0 1 3 1 】

まず、チューニング処理部 2 5 は、プログラマによる目標処理時間 $(\tau)_T$ 、繰返回数 ic_{max} 及び制限並列効率 $(E_p)_{max}$ の設定入力を受け付ける（ステップ S 9 1）。次に、ログデータ格納部 3 0 に格納されている並列効率及び処理時間のデータ（例えばチューニングしようとしているプログラムの処理ログに含まれる並

列効率及び処理時間)を用いて、目標処理時間 $(\tau)_T$ に対応する目標並列効率 $(E_p)_T$ を計算し、記憶装置に格納する(ステップS93)。目標並列効率 $(E_p)_T$ は以下の式で計算される。この式は線形外挿を表している。

$$(E_p)_T = \max(\tau_i) \times E_p(p) / (\tau)_T$$

【0132】

そして目標並列効率 $(E_p)_T$ が制限並列効率 $(E_p)_{\max}$ 以下であるか判断する(ステップS95)。目標処理時間 $(\tau)_T$ を何らの制限なく設定すると、実現不可能な目標並列効率 $(E_p)_T$ が設定されることになりかねないため、本ステップにおいて目標処理時間 $(\tau)_T$ が妥当であるか判断するものである。もし、目標並列効率 $(E_p)_T$ が制限並列効率 $(E_p)_{\max}$ を超える場合には、目標処理時間 $(\tau)_T$ 又は制限並列効率 $(E_p)_{\max}$ の設定し直しが必要となるので、ステップS91に戻る。

【0133】

一方、目標並列効率 $(E_p)_T$ が制限並列効率 $(E_p)_{\max}$ 以下である場合には、今回測定 of 処理時間 $\tau(p)$ が目標処理時間 $(\tau)_T$ 以下になっているか判断する(ステップS97)。なお、ステップS97の最初の処理は、必ずNと判断される。もし、今回測定 of 処理時間が目標処理時間 $(\tau)_T$ 以下になっている場合には、目標を達成した旨のメッセージ、達成された並列効率、処理時間 $\tau(p)$ 等のデータを表示装置等の出力装置110に出力する(ステップS99)。一方、今回測定 of 処理時間が目標処理時間 $(\tau)_T$ を超えている場合には、カウンタ値 i_c が繰返回数 i_{\max} 以上になっているか判断する(ステップS101)。もし、カウンタ値 i_c が繰返回数 i_{\max} 以上になった場合には、目標達成不可能を表すメッセージ、達成できた並列効率、処理時間 $\tau(p)$ 等のデータを、表示装置などの出力装置110に出力する(ステップS103)。

【0134】

もし、カウンタ値 i_c が繰返回数 i_{\max} 未満である場合には、カウンタ値 i_c を1インクリメントする(ステップS105)。そして、冗長処理、ロードバランス、通信処理、I/Oなどの並列性能阻害要因についてチューニングを行う(ステップS107)。プログラムの書き換えではなく、ツールやコンパイラ、ランタイムライブラリなどによりチューニングを実施しても良い。プログラマが実施

する作業である場合もあるのでここでは点線ブロックで示している。チューニングの後に、並列計算機システム 2 0 0 にてプログラムを再度並列処理し、同時に測定部 2 0 1 により処理時間等の測定処理を実施し、記憶装置に格納する（ステップ S 1 0 9）。ステップ S 1 0 9 も並列性能分析装置 1 0 0 の処理ではないので一点鎖線によるブロックで表している。この後、データ取得部 1 0 が、並列計算機システム 2 0 0 から処理時間等のデータを取得し、ログデータ格納部 3 0 に格納する。そして、ロードバランス寄与率計算部 1 1、仮想並列化率計算部 1 2、並列性能阻害要因寄与率計算部 1 3、並列効率計算部 1 4 などにより、並列効率を含む並列性能評価指標を計算し、ログデータ格納部 3 0 に格納する（ステップ S 1 1 1）。そして、ステップ S 9 7 に戻る。

【 0 1 3 5 】

このように、所定のチューニング回数だけ目標処理時間 $(\tau)_T$ を達成すべくチューニング作業を実施することになるので、プログラマも効率的な作業を実施することができるようになる。

【 0 1 3 6 】

例えば図 2 3 のような処理時間を基に具体例を示しておく。この際、 $\tau(p) = 37$ 、 $E_p(4) = 0.4443$ であるので、今仮に $(E_p)_{\max} = 0.6$ 、 $(\tau)_T = 28$ とすると、 $(E_p)_T = 0.5871$ となる。従って、ステップ S 9 5 からステップ S 9 7 に移行する。最初の処理であるからステップ S 9 7 からステップ S 1 0 1 及び S 1 0 5 を介してステップ S 1 0 7 に移行する。そこで 1 回目のチューニングとして通信時間 x_C を $1/2$ に削減したものとする。その結果を用いてステップ S 1 1 1 で並列効率を含む並列性能評価指標を計算する。そうすると図 3 2 に示すような結果が得られる。なお、図 3 2 は処理時間 $\max(\tau_i)$ を加えて比較したものである。

【 0 1 3 7 】

チューニングとして通信時間 x_C を $1/2$ に削減した場合の計算方法は以下のとおりである。なお、 $x_{1,C}(4) = 10/2 = 5$ 、 $x_{2,C}(4) = 11/2 = 5.5$ 、 $x_{3,C}(4) = 12/2 = 6$ 、 $x_{4,C}(4) = 13/2 = 6.5$ とする。また式 (12-1) から以下のように計算される。

【数 6 4】

$$\chi_{1,RED}(1) \equiv \frac{1}{p} \cdot \sum_{i=1}^p \chi_{i,RED}(p) = \frac{1}{4} \cdot (8+9+7+7) = 7.75$$

(1) ロードバランス寄与率 (式 (5))

【数 6 5】

$$\begin{aligned} R_b(p) &= \frac{\sum_{i=1}^p \tau_i(p)}{\tau(p) \cdot p} = \frac{\sum_{i=1}^p (\gamma_i(p) + \chi_{i,RED}(p) + \chi_{i,C}(p) + \chi_{i,others}(p))}{\max(\tau_i(p)) \cdot p} \\ &= \frac{(15+8+1)+5+(14+9+1)+5.5+(13+7+1)+6+(16+7+1)+6.5}{(16+7+13/2+1) \cdot 4} \\ &= \frac{29+29.5+27+30.5}{30.5 \cdot 4} = \frac{116}{122} \\ &= 0.9508 \end{aligned}$$

(2) 仮想並列化率 (式 (6-1))

【数 6 6】

$$R_p(p) = \frac{\sum_{i=1}^p \gamma_i(p)}{\tau(1)} = \frac{(15+14+13+16)}{(15+14+13+16)+7.75} = \frac{58}{(58+7.75)} = 0.8821$$

(3) 並列性能阻害要因寄与率 (式 (7))

【数 6 7】

$$R_{RED}(p) = \frac{\sum_{i=1}^4 \chi_{i,RED}(p)}{\sum_{i=1}^4 \tau_i(p)} = \frac{(8+9+7+7)}{116} = 0.2672$$

$$R_C(p) = \frac{\sum_{i=1}^p \chi_{i,C}(p)}{\sum_{i=1}^p \tau_i(p)} = \frac{5+5.5+6+6.5}{116} = 0.1983$$

$$R_{Others}(p) = \frac{\sum_{i=1}^p \chi_{i,others}(p)}{\sum_{i=1}^p \tau_i(p)} = \frac{1+1+1+1}{116} = 0.0345$$

(4-1) 並列効率 (式 (4-4))

【数 6 8】

$$\begin{aligned} E_p(p) &= R_b(p) \cdot \frac{1}{R_p(p)} \cdot (1 - R_{RED}(p) - R_C(p) - R_{Others}(p)) \\ &= 0.9508 \cdot \frac{1}{0.8821} \cdot (1 - 0.2672 - 0.1983 - 0.0345) = 0.5389 \end{aligned}$$

(4-2) 並列効率 (式 (9-2))

【数 6 9】

$$E_p(4) = R_b(4) \cdot \frac{1}{R_p(4)} \cdot \frac{\sum_{i=1}^p \gamma_i(4)}{\sum_{i=1}^p \tau_i(4)} = 0.9508 \cdot \frac{1}{0.8821} \cdot \frac{58}{116} = 0.5389$$

【 0 1 3 8 】

1 回のチューニングでは処理時間 $\max(\tau_i)$ ($= \tau(p)$) は 3 0 . 5 で目標処理時間 $(\tau)_T$ を達成できていないので、再度何らかのチューニングを実施する必要がある。

【 0 1 3 9 】

従来、並列性能の評価は、プロセッサ数を変えての時間変化、他のシステムとの処理時間比較、時間内に行われたオペレーション数の比較等、処理時間比較をベースにして行われた。これには 2 度以上の時間測定が必要で、プログラム開発時間を増加させる原因となっていた。またこの比較という相対的な並列性能評価では、処理データが変わった場合、比較基準を再度測定する必要がある。このように並列性能評価に時間がかかる結果、ある条件でしか並列性能が出ないアプリケーション・プログラムが開発されてしまう場合が生ずる。上で述べたような処理を実施することにより、並列効率による並列性能評価が 1 回の測定でできるようになり、並列アプリケーション・プログラムの開発時間のうちの性能評価時間を大幅に短縮することが可能となる。その結果、並列性能を十分考慮した並列アプリケーション・プログラムの開発が現実的に実施できるようになる。

【 0 1 4 0 】

また従来では、アプリケーション・プログラムのチューニングによる性能向上作業は達成目標が不明確であったため作業時間見積が容易でなかった。またどのような場合に作業を終えるかが明確にならず、結果的に多大な作業時間を費やす場合も生じていた。さらにアプリケーション・プログラムのチューニングではなく再開発が必要になってしまう場合もあった。上で述べたような処理を実施することにより、アプリケーション・プログラムのチューニングによる並列効率向上の目標を明確に定め、チューニングの繰返回数等で作業時間の予測もできるようになる。

【 0 1 4 1 】

さらに従来では、アプリケーション・プログラムのチューニングはアプリケーション・プログラムの中で処理時間が長い手続き（アプリケーション・プログラムの一部分）を時間測定等により探し出し、その手続き内で問題となっている並

列性能阻害要因を処理時間の比較により探し出し、その処理時間を減らすという形で行われた。上で述べたような処理により、このようなアプリケーション・プログラムの一部分のチューニングに対し、ロードバランスに対する性能評価が初めて可能となった。

【 0 1 4 2 】

F. アルゴリズム選定処理

従来、並列アプリケーション・プログラムの一部に用いるアルゴリズムを変えたものの同士の性能比較には処理時間を用いたが、処理時間減少の原因が並列処理の効果によるものか、機能の違いによる効果か（例えば演算数の減少か）判別できなかった。その結果、処理時間は短いがスケーラビリティが悪いアルゴリズムにたくさんのプロセッサをつぎ込む資源の無駄使いを見逃すこととなった。本実施の形態では、より並列効率の良いアルゴリズムを選定して、システム全体の運用効率を向上させる。ここではまず、並列処理に向かないアルゴリズムと並列処理に向くアルゴリズムを比較した例を示しておく。

【 0 1 4 3 】

[並列処理に向かないアルゴリズム]

例えば図 3 3 に示すような処理時間の測定がなされた場合を例に説明する。なお、 $\chi_{1,C}(1) = 0$ とする。また式 (1 2 - 1) から以下のように計算される。

【数 7 0】

$$\chi_{1,RED}(1) \equiv \frac{1}{P} \cdot \sum_{i=1}^P \chi_{i,RED}(P) = \frac{1}{4} \cdot (50 + 50 + 50 + 50) = 50$$

(1) ロードバランス寄与率 (式 (5))

【数 7 1】

$$\begin{aligned}
 R_b(p) &= \frac{\sum_{i=1}^p \tau_i(p)}{\tau(p) \cdot p} \\
 &= \frac{4 \cdot (50 + 50 + 10)}{(50 + 50 + 10) \cdot 4} \\
 &= 1
 \end{aligned}$$

(2) 仮想並列化率 (式 (6-1))

【数 7 2】

$$R_p(p) = \frac{\sum_{i=1}^p \gamma_i(p)}{\tau(1)} = \frac{(50 + 50 + 50 + 50)}{(50 + 50 + 50 + 50) + 50} = \frac{200}{(200 + 50)} = 0.8000$$

(3) 加速率 (式 (6-2))

【数 7 3】

$$A_p(p) = \frac{1}{1 - R_p(p)} = \frac{1}{1 - 0.8000} = 5.000$$

(4) 並列性能阻害要因寄与率 (式 (7))

【数 7 4】

$$R_{RED}(p) = \frac{\sum_{i=1}^4 \chi_{i,RED}(p)}{\sum_{i=1}^4 \tau_i(p)} = \frac{(50+50+50+50)}{440} = 0.4545$$

$$R_C(p) = \frac{\sum_{i=1}^p \chi_{i,C}(p)}{\sum_{i=1}^p \tau_i(p)} = \frac{10+10+10+10}{440} = 0.09091$$

(4-1) 並列効率 (式 (4-4))

【数 7 5】

$$\begin{aligned} E_p(p) &= R_b(p) \cdot \frac{1}{R_p(p)} \cdot (1 - R_{RED}(p) - R_C(p)) \\ &= 1.000 \cdot \frac{1}{0.8000} \cdot (1 - 0.4545 - 0.09091) = 0.5682 \end{aligned}$$

(4-2) 並列効率 (式 (9-1))

【数 7 6】

$$E_p(4) = R_b(4) \cdot \frac{1}{R_p(4)} \cdot \frac{\sum_{i=1}^p \gamma_i(4)}{\sum_{i=1}^p \tau_i(4)} = 1.000 \cdot \frac{1}{0.8000} \cdot \frac{200}{440} = 0.5682$$

【0 1 4 4】

[並列処理に向くアルゴリズム]

図 3 4 に示すような処理時間が測定された場合の例を説明する。なお、 $\chi_{1,C}(1) = 0$ とする。また式 (1 2-1) から以下のように計算される。

【数 7 7】

$$\chi_{1,RED}(1) \equiv \frac{1}{p} \cdot \sum_{i=1}^p \chi_{i,RED}(p) = 0$$

(1) ロードバランス寄与率 (式 (5))

【数 7 8】

$$\begin{aligned} R_b(p) &= \frac{\sum_{i=1}^p \tau_i(p)}{\tau(p) \cdot p} \\ &= \frac{4 \cdot (110 + 10)}{(110 + 10) \cdot 4} \\ &= 1 \end{aligned}$$

(2) 仮想並列化率 (式 (6-1))

【数 7 9】

$$R_p(p) = \frac{\sum_{i=1}^p \gamma_i(p)}{\tau(1)} = \frac{110 + 110 + 110 + 110}{110 + 110 + 110 + 110} = 1.000$$

(3) 加速率 (式 (6-2))

【数 8 0】

$$A_p(p) = \frac{1}{1 - R_p(p)} = \frac{1}{1 - 1} = \infty$$

(4) 並列性能阻害要因寄与率 (式 (7))

【数 8 1】

$$R_{RED}(p) = \frac{\sum_{i=1}^4 \chi_{i,RED}(p)}{\sum_{i=1}^4 \tau_i(p)} = \frac{0+0+0+0}{480} = 0.0000$$

$$R_C(p) = \frac{\sum_{i=1}^p \chi_{i,C}(p)}{\sum_{i=1}^p \tau_i(p)} = \frac{10+10+10+10}{480} = 0.08333$$

(4-1) 並列効率 (式 (4-4))

【数 8 2】

$$E_p(p) = R_b(p) \cdot \frac{1}{R_p(p)} \cdot (1 - R_{RED}(p) - R_C(p))$$

$$= 1.000 \cdot \frac{1}{1.000} \cdot (1 - 0.0000 - 0.08333) = 0.9167$$

(4-2) 並列効率 (式 (9-1))

【数 8 3】

$$E_p(4) = R_b(4) \cdot \frac{1}{R_p(4)} \cdot \frac{\sum_{i=1}^p \gamma_i(4)}{\sum_{i=1}^p \tau_i(4)} = 1.000 \cdot \frac{1}{1.000} \cdot \frac{440}{480} = 0.9167$$

【0145】

以上の処理結果をまとめると図 35 のようになる。並列処理に向かないアルゴリズムの番号を $j = 1$ 、並列処理に向くアルゴリズムの番号を $j = 2$ とすると、 $j = 1$ では加速率 $A_p = 5.000$ で有限であり、プロセッサを増加させても 5

個分が効率的に限度であることが分かる。一方、 $j = 2$ では加速率 $A_p = \infty$ であり、プロセッサを投入すればするほど処理時間が短くなる可能性がある。なお、処理時間 τ は $j = 1$ の方が110で $j = 2$ の120より短いので、今まであれば並列処理に向かないアルゴリズムである $j = 1$ を選択してしまう場合があった。

【0146】

そこで本実施の形態では図36に示す処理をアルゴリズム選定処理部26にて実施するものとする。まず、プログラマにより目標処理時間 $(\tau)_T$ の設定入力を受け付ける(ステップS121)。そして、初期設定として、アルゴリズム番号 j を1に、最適なアルゴリズム番号 j_T を1に設定する(ステップS123)。また、 $j = 1$ の場合に、目標処理時間 $(\tau)_T$ を達成するために必要なプロセッサ数 $(p)_1$ を線形外挿で計算する(ステップS125)。すなわち、ログデータ格納部30に格納されたアルゴリズム番号 $j = 1$ の処理時間等を用いて、 $(p)_1 = (\tau)_1 / (\tau)_T / (E_p)_1 + (p)_1$ を計算し、記憶装置に格納する(ステップS125)。また、最適なアルゴリズムについて必要なプロセッサ数 $(p)_T = \text{INT}((p)_1 + 0.99)$ と設定する。さらに、 $p_{\min} = p_1$ と設定する。

【0147】

次に j を1インクリメントする(ステップS127)。そして、 j の場合のプロセッサ数 $(p)_j$ を以下の式で計算し、記憶装置に格納する(ステップS129)。

$$(p)_j = (\tau)_j / (\tau)_T / (E_p)_j + (p)_j$$

そして、 $(p)_j < (p)_{\min}$ であり、且つ $(A_p)_j > (p)_j$ であるか確認する(ステップS131)。すなわち、 $(p)_j$ が最小であり、当該アルゴリズムの加速率 $(A_p)_j$ より最適プロセッサ数が小さくなっているか、すなわち実現可能かを確認する。ステップS125及びS129では線形外挿で単純に $(p)_j$ を計算しているので、実現可能か否かをここで担保するものである。もし、 $(p)_j < (p)_{\min}$ であり、且つ $(A_p)_j > (p)_j$ である場合には、アルゴリズム番号 j を j_T に設定する。すなわち、 $j_T = j$ 。また、 $(p)_T = \text{INT}((p)_j + 0.99)$ と設定する(ステップS133)。

【0148】

ステップ S 1 3 1 又はステップ S 1 3 3 の後に、 j がアルゴリズム数 j_{\max} 以上になっているか確認する（ステップ S 1 3 5）。すなわち全てのアルゴリズムについて処理したか判断する。もし、 $j \geq j_{\max}$ であれば、最終的に j_T で特定されたアルゴリズム番号及びその場合のプロセッサ数 $(p)_T$ 並びに他の処理結果（ j , $(p)_j$, $(A_p)_j$, $(\tau)_j$ 等のセット）を、表示装置等の出力装置 1 1 0 に出力する（ステップ S 1 3 7）。一方、 $j < j_{\max}$ であればステップ S 1 2 7 に戻る。

【 0 1 4 9 】

このようにすれば実現性のある範囲内でプロセッサ数が少なく目標処理時間を達成することができるアルゴリズムを特定することができる。また、本処理フローで最適とされたアルゴリズムだけではなく、あまりプロセッサ数が異なるアルゴリズムでチューニングなどがやりやすいアルゴリズムを選択することもできる。

【 0 1 5 0 】

図 3 5 に示した 2 つのアルゴリズムの例で具体的に説明する。初めに目標処理時間 $(\tau)_T = 50$ を設定する。次にそれらのアルゴリズムの $(E_p)_j$, $(\tau)_j$ を用いて線形外挿により必要なプロセッサ数 $(p)_j$ を計算する。図 3 6 に示した処理フローでは $(p)_j$ を線形外挿で求めるため、冗長処理のみを考慮した $A_p(4)$ をプロセッサ数の上限として導入して、 $A_p(4) > (p)_j$ であれば $(p)_j$ を適用できるものとする。その結果、並列処理に向かないアルゴリズムの限界性能が 5.000 であるのに対して $(p)_j$ は 7.872 で、並列処理に向かないアルゴリズムについては $(p)_j$ を適用できないことが分かる。一方、並列処理に向くアルゴリズムの限界性能は ∞ であるので、6.618 のプロセッサで $(\tau)_T = 50$ を達成できる可能性がある。従って並列処理に向くアルゴリズム $j_T = 1$ を選ぶことができる。その場合の初めの目安は、6.618 を切り上げて得た $(p)_T = 7$ である。今までは τ の 1 1 0 と 1 2 0 を比べて処理時間が短いが並列処理に向かないアルゴリズムを採用する場合が多かったが、図 3 6 の処理フローにより $p = 4$ では処理時間が長いが並列処理に向くアルゴリズムを選択できるようになる。

【 0 1 5 1 】

G. 並列性能評価処理

本実施の形態においては、実運用における全処理の並列性能評価指標のログデータを作成することが可能となる。このログデータにおいてある特定の処理をターゲットにすれば専用並列計算機システムに必要な仕様書（CPU性能、通信性能、I/O性能、ランタイムライブラリの性能等）を求めることが可能となる。全アプリケーションによる処理をターゲットにすればそのログに対する汎用並列並列計算機システムに必要な仕様書を作成することも可能となる。

【0152】

例えば図37に示した処理番号1乃至4の処理性能を向上するためには、通信性能を上げること、又はCPUの性能と通信の性能の比を保つ形で両者の性能向上を図れば良いことが分かる。並列性能評価処理部27は、例えばログデータ格納部30に格納されているデータから図37のようなテーブルを構成し、表示装置等の出力装置110に出力する。また、並列性能阻害要因のうちどの処理においても相対的に高い値を示しているものを強調表示するような処理を行っても良い。また、処理5の性能を向上するためには、システムのリプレイス等による性能向上ではなく、アプリケーション・プログラムのチューニング等が必要なことが分かる。これは、処理5だけ冗長処理による並列性能阻害要因寄与率が大きな値を示しているからであり、並列性能評価処理部27は特徴的な処理についても抽出して、例えば強調表示させるような場合もある。

【0153】

通信性能を決定する方法としては、例えばシステムリプレイスデータ処理で説明したような処理を実施すればよい。すなわち、通信性能の倍率以外は1に固定してしまい、目標の $E_p(4)$ をクリアするまで実施する。

【0154】

なお、処理数の多い処理番号1乃至4のパターンの処理に着目して通信性能を向上すれば、このシステムは汎用並列計算機システムとなる。一方、処理数の少ない処理番号5のパターンの処理に着目して冗長処理を軽減する仕組みを計算機システムに導入すれば、専用並列計算機システムとなる。また、処理番号5のアプリケーション・プログラムのチューニングを行い、冗長処理を減らせば、通信

性能を向上するだけで 1 乃至 5 の汎用並列計算機システムとなる。

【 0 1 5 5 】

従来では、アプリケーション・プログラムの並列処理の特徴により並列計算機システムの並列性能が大きく変わるため、並列計算機システムを開発することが容易でなかった。それを克服する方法として、アプリケーション・プログラムを特定し並列性能を分析して、それに合った並列計算機システムを開発する方法が多く用いられていた。しかしこの方法ではアプリケーション・プログラムが変わると、全く並列性能を発揮できないシステムを開発してしまう恐れがある。本実施の形態によれば、実運用における全処理の並列性能評価指標のログデータを作成することが可能となるため、このログデータを基にして、ある特定の処理をターゲットにすれば専用並列計算機システムに必要な仕様書（CPU性能、通信性能、I/O性能、ランタイムライブラリの性能等）を作成することが可能となる。また、全処理をターゲットにすればそのログに対する汎用並列計算機システムに必要な仕様書を作成することも可能となる。

【 0 1 5 6 】

また従来では、並列計算機システムの並列性能阻害要因を定量的に把握する手段が組み込まれているか否かはシステムにより大きく差があり、並列性能阻害要因を定量的に把握する手段を全く持っていないシステムもある。本実施の形態では、式（7）で示すように、並列性能阻害要因が無い状態から任意に要因を追加できる機能を持つため、販売後、システムのアプグレード時に要因測定機能を追加して評価精度を高めることができる。

【 0 1 5 7 】

さらに、従来の性能評価指標である例えばflop/s, Mop/s, tpmC等は、アプリケーション・プログラムの種類によって適用できるものとできないものがある。本実施の形態では、指標を時間比で表わすため、全てのアプリケーション・プログラムに対して有効であり、性能評価を適切に実施できるようになる。さらに、従来の並列性能評価方法には特別な並列処理にしか適用できないものがあつたが、本実施の形態によれば全ての並列処理に適用することができる。

【 0 1 5 8 】

以上本発明の実施の形態を説明したが、これにより、並列処理の性能を表わす並列効率に対し、それを低下させる割合を並列性能評価指標、すなわちロードバランス寄与率、仮想並列化率及び並列性能阻害要因で示すことができるようになる。並列性能評価指標にロードバランス寄与率が加わり、全ての並列処理の並列性能評価が可能となる。

【 0 1 5 9 】

また、式（８－２）を用いれば、 $R_p(p)$ がほぼ１である場合には、並列効率の計算に推測値 $\tau(1)$ を必要としないため、 $\tau(1)$ を測定できないグリッドやクラスタによる並列処理を含め、全ての並列処理の正確な（推測値 $\tau(1)$ を含まないと言う意味で）並列性能評価が可能となる。

【 0 1 6 0 】

さらに、式（９－１）及び式（９－２）を用いれば、 $R_p(p) < 1$ の場合でも、並列効率の計算のために $\tau(1)$ を見積ることにより、 $\tau(1)$ を測定できないグリッドやクラスタによる並列処理を含め、全ての並列処理の並列性能評価が可能となる。

【 0 1 6 1 】

式（７）の式の形により、対象とする並列計算機システムに特有の並列処理阻害要因を随時導入可能となり、詳細な性能評価を容易に実施できる。さらに、並列性能阻害要因の寄与率を並列効率に対する百分率で捉えられ、直感的な並列性能評価が可能となる。

【 0 1 6 2 】

また、ロードバランスの寄与が、並列効率に対する比率という数値で明確となったため、今まで評価できなかった並列性能に対するロードバランスの寄与が具体的に示せるようになる。

【 0 1 6 3 】

また、並列性能指標を計算して提示するだけでなく、処理時間測定により決定された並列効率を用い、効率の良い処理を行うプロセッサ数の決定ができる。さらに、並列処理の効率を考慮した上で、プロセッサの増減を検討できる。

【 0 1 6 4 】

さらに、性能仕様が異なる新しい並列計算機システムの導入を机上で検討できる。また、並列性能評価指標を用いて、システム運用における利用効率管理ができる。

【 0 1 6 5 】

以上本発明の実施の形態を説明したが、本発明はこれに限定されない。例えば図 1 9 の機能ブロック図は一例であり、必ずしもプログラムモジュールとは対応しない。また、プロセッサ数最適化処理部 2 1、プロセッサ増設見積処理部 2 2、システムリプレイスデータ処理部 2 3、運用効率データ処理部 2 4、チューニング処理部 2 5、アルゴリズム選定処理部 2 6、並列性能評価処理部 2 7 は全て備えていなければならないものではなく、全て設けられる場合もあれば全く設けられない場合もある。さらに、任意の組み合わせにて設けられる場合もある。

【 0 1 6 6 】

〔実施例〕

上で述べた実施の形態は全ての並列処理（メモリ、ネットワーク、CPU性能が同じホモ構成又は異なるヘテロ構成のグリッド、クラスタ又は分散メモリ、若しくはSMP（対称型マルチプロセッシング：Symmetric MultiProcessing）、SMP＋分散メモリ、NUMA（NonUniform Memory Access）等）に適用可能である。以下では、代表的な態様について計算例を示しておく。

【 0 1 6 7 】

(1) ホモ構成におけるグリッド等 ($x_{1,j}(1) = 0$)

グリッドやクラスタで処理を行う場合、各プロセッサへの処理の割り付けや処理結果の回収にネットワークを用いるため通信が発生するが、これは1つのプロセッサで処理する場合には生じない。このような処理は $x_{1,j}(1) = 0$ の処理である。ここでは並列性能阻害要因を通信のみとし、 $x_{1,c}(1) = 0$ であるような処理の並列性能を評価する。例えば図 3 8 のような経過時間の測定結果が得られた場合について説明する。

【 0 1 6 8 】

式 (3) から以下の計算がなされる。

【数 8 4】

$$\tau_1(4) = 5 + 25 + 15 + 15 + 10 = 70$$

$$\tau_2(4) = 10 + 20 + 10 + 10 + 10 = 60$$

$$\tau_3(4) = 15 + 15 + 10 + 15 + 10 = 65$$

$$\tau_4(4) = 10 + 10 + 10 + 25 + 10 = 65$$

式 (5)、式 (6-1)、式 (6-2) 及び式 (7) からそれぞれ以下のような計算がなされる。

【数 8 5】

$$R_b(4) = \frac{70 + 60 + 65 + 65}{70 \times 4} = 0.9286$$

$$R_p(4) = \frac{40 + 30 + 30 + 35}{40 + 30 + 30 + 35} = 1.000$$

$$A_p(4) = \frac{1}{1-1} = \infty$$

$$R_C(4) = \frac{30 + 30 + 35 + 30}{260} = 0.4808$$

並列効率については、順番に式 (4-4)、式 (4-5)、式 (8-2)、式 (9-1)、式 (9-2) からそれぞれ以下のような計算がなされる。

【数 8 6】

$$E_p(4) = 0.9286 \times \frac{1}{1.000} \times (1 - 0.4808) = 0.4821$$

$$E_p(4) = 0.9286 \times \frac{1}{1 - 1/\infty} \times (1 - 0.4808) = 0.4821$$

$$E_p(4) = 0.9286 \times (1 - 0.4808) = 0.4821$$

$$E_p(4) = 0.9286 \times \frac{1}{1.000} \times \frac{40 + 30 + 30 + 35}{70 + 60 + 65 + 65} = 0.4822$$

$$E_p(4) = ((40 + 30 + 30 + 35) + 0) / (70 \times 4) = 0.4821$$

【0 1 6 9】

以上計算された並列性能評価指標をまとめると図 3 9 のようになる。 $A_p(p) = \infty$ ゆえ $p = \infty$ で並列処理した時の性能向上の可能性は無限であるが、プロセッサ数 $p = 4$ を投入した現実の性能向上 $E_p(4) \cdot p$ は 1.928 である。その理由は、並列効率 $E_p(4)$ が、ロードバランス寄与率で 93% ($Rb(4) = 0.9286$) となり、通信によりさらに 48% ($Rc(4) = 0.4808$) 低下するためである。

【0 1 7 0】

(2) ホモ構成におけるグリッド等 ($x_{1,RED}(1) \neq 0$)

数値計算では、アプリケーション・プログラムを全てのプロセッサにコピーし、ループ処理のインデックス等を識別して各プロセッサで処理を分担する、いわゆるデータパラレルで並列処理する場合が多い。データパラレルでは、例えばループ間に並列処理できない処理が残る。この処理を全プロセッサが行うとき、内容が同じ処理であることからこれを冗長処理と呼ぶ。冗長処理の特徴は、並列処理でない場合も必要な処理のため必ず $x_{1,RED}(1) \neq 0$ となることである。ここでは並列性能阻害要因を冗長処理のみとし、 $x_{1,RED}(1) \neq 0$ であるような処理の並列性能を評価する。例えば図 4 0 のような経過時間の測定結果が得られた場合について説明する。

【0 1 7 1】

式 (3) から以下の計算がなされる。

【数 8 7】

$$\tau_1(4) = 8 + 35 + 10 + 20 + 5 = 78$$

$$\tau_2(4) = 10 + 33 + 11 + 22 + 7 = 83$$

$$\tau_3(4) = 7 + 37 + 10 + 19 + 4 = 77$$

$$\tau_4(4) = 11 + 30 + 9 + 18 + 6 = 74$$

式 (5)、式 (12-1)、式 (6-1)、式 (6-2) 及び式 (7) からそれぞれ以下のような計算がなされる。

【数 8 8】

$$R_b(4) = \frac{78 + 83 + 77 + 74}{83 \times 4} = 0.9398$$

$$\chi_{1,RED}(1) = \frac{1}{4} \cdot (23 + 28 + 21 + 26) = 24.5$$

$$R_p(4) = \frac{55 + 55 + 56 + 48}{(55 + 55 + 56 + 48) + 24.5} = 0.8973$$

$$A_p(4) = \frac{1}{1 - 0.8973} = 9.737$$

$$R_{RED}(4) = \frac{23 + 28 + 21 + 26}{78 + 83 + 77 + 74} = 0.3141$$

並列効率については、順番に式 (4-4)、式 (4-5)、式 (9-1)、式 (9-2) からそれぞれ以下のような計算がなされる。

【数 8 9】

$$E_p(4) = 0.9398 \times \frac{1}{0.8973} \times (1 - 0.3141) = 0.7184$$

$$E_p(4) = 0.9398 \times \frac{1}{1 - 1/9.737} \times (1 - 0.3141) = 0.7184$$

$$E_p(4) = 0.9398 \times \frac{1}{0.8973} \times \frac{55 + 55 + 56 + 48}{78 + 83 + 77 + 74} = 0.7184$$

$$E_p(4) = ((55 + 55 + 56 + 48) + 24.5) / (83 \times 4) = 0.7184$$

【0 1 7 2】

以上計算された並列性能評価指標をまとめると図 4 1 のようになる。ここでは $A_p(p) = 9.737$ ゆえ $p > 9$ の並列処理は無意味である。プロセッサ数 $p = 4$ を投入した現実の性能向上 $E_p \cdot p$ は 2.874 である。その理由は、並列効率 E_p が、ロードバランス寄与率で 94% ($R_b(4) = 0.9398$) となり、冗長処理によりさらに 31% ($R_{RED}(4) = 0.3141$) 低下するためである。

【0 1 7 3】

(3) ホモ構成におけるグリッド等 ($x_{1,j}(1) \neq 0$: 但し冗長処理以外)

例えば通信ライブラリの処理時間はネットワーク通信と演算で構成される。この演算時間を $x_{1,C}(1)$ として扱う。ここでは並列性能阻害要因を通信のみとし、 $x_{1,C}(1) \neq 0$ であるような処理の並列性能を評価する。例えば図 4 2 のような経過時間の測定結果が得られた場合について説明する。

【0 1 7 4】

式 (3) から以下の計算がなされる。

【数 9 0】

$$\tau_1(4) = 2 + 5 + 25 + 2 + 15 + 15 + 2 + 10 = 76$$

$$\tau_2(4) = 3 + 10 + 20 + 2 + 10 + 10 + 2 + 10 = 67$$

$$\tau_3(4) = 2 + 15 + 15 + 2 + 10 + 15 + 2 + 10 = 71$$

$$\tau_4(4) = 2 + 10 + 10 + 2 + 10 + 25 + 2 + 10 = 71$$

式 (5)、式 (12-1)、式 (6-1)、式 (6-2) 及び式 (7) からそれぞれ以下のような計算がなされる。

【数 9 1】

$$R_b(4) = \frac{76 + 67 + 71 + 71}{76 \times 4} = 0.9375$$

$$\chi_{1,C}(1) = \frac{1}{4} \cdot (6 + 7 + 6 + 6) = 6.25$$

$$R_p(4) = \frac{40 + 30 + 30 + 35}{(40 + 30 + 30 + 35) + 6.25} = 0.9557$$

$$A_p(4) = \frac{1}{1 - 0.9557} = 22.6$$

$$R_C(4) = \frac{36 + 37 + 41 + 36}{285} = 0.5263$$

並列効率については、順番に式 (4-4)、式 (4-5)、式 (9-1)、式 (9-2) からそれぞれ以下のような計算がなされる。

【数 9 2】

$$E_p(4) = 0.9375 \times \frac{1}{0.9557} \times (1 - 0.5263) = 0.4647$$

$$E_p(4) = 0.9375 \times \frac{1}{1 - 1/22.6} \times (1 - 0.5263) = 0.4647$$

$$E_p(4) = 0.9375 \times \frac{1}{0.9557} \times \frac{40 + 30 + 30 + 35}{76 + 67 + 71 + 71} = 0.4647$$

$$E_p(4) = ((40 + 30 + 30 + 35) + 6.25) / (76 \times 4) = 0.4646$$

【0 1 7 5】

以上計算された並列性能評価指標をまとめると図 4 3 のようになる。ここでは $A_p(p) = 22.57$ ゆえ $p < 22$ で処理すべきである。プロセッサ数 $p = 4$ を投入した現実の性能向上 $E_p \cdot p$ は 1.859 である。その理由は、並列効率 E_p が通信により 53% ($R_c = 0.5263$) 低下するためである。ロードバランス寄与率は 94% ($R_b(4) = 0.9375$) で、ロードバランスはこの場合の並列性能を阻害する主要な要因ではない。実施例 (1) と異なるところは、 $x_{1,C}(1) \neq 0$ のため $A_p(p)$ が有限値になるところである。

【0 1 7 6】

通信処理に含まれる演算はプロセッサの増加と共に変化する可能性がある。これを $X_{i,C}(p)$ とみなして並列性能評価に取り入れることで、プロセッサ数によって異なる演算数を評価に組み込むことが可能となる。

【0 1 7 7】

(4) ホモ構成におけるグリッド等 (待ち (アイドリング: ウェイト (wait) と呼ぶ) がある場合)

特定のプロセッサが処理し、結果を他のプロセッサが使う場合、その処理が終了するまで他のプロセッサは次の処理を開始できない。例えば特定のプロセッサのみがデータベース (DB) をアクセスできる場合がこれに当たる。図 4 4 ではプロセッサ # 1 でこの処理 (γ') を行う。他のプロセッサは DB 処理の間、待

ち状態になる。このようにCPUを待たせておくアイドル処理が存在する場合の並列性能を評価することができる。図44のような経過時間の測定結果が得られたものとする。

【0178】

式(3)から以下の計算がなされる。

【数93】

$$\tau_1(4) = 5 + 25 + 10 + 10 + 3 + 35 + 5 = 93$$

$$\tau_2(4) = 10 + 20 + 10 + 10 + 5 + 30 + 4 = 89$$

$$\tau_3(4) = 15 + 15 + 10 + 10 + 4 + 33 + 3 = 90$$

$$\tau_4(4) = 10 + 10 + 10 + 20 + 6 + 29 + 4 = 89$$

式(5)、式(12-1)、式(6-1)、式(6-2)及び式(7)からそれぞれ以下のような計算がなされる。

【数94】

$$R_b(4) = \frac{93 + 89 + 90 + 89}{93 \times 4} = 0.9704$$

$$R_p(4) = \frac{80 + 50 + 48 + 39}{80 + 50 + 48 + 39} = 1.000$$

$$A_p(4) = \frac{1}{1-1} = \infty$$

$$R_C(4) = \frac{23 + 29 + 32 + 30}{361} = 0.3158$$

$$R_W(4) = \frac{0 + 10 + 10 + 20}{361} = 0.1108$$

並列効率については、順番に式(4-4)、式(4-5)、式(8-2)、式(9-1)、式(9-2)からそれぞれ以下のような計算がなされる。

【数 9 5】

$$E_p(4) = 0.9704 \times \frac{1}{1.000} \times (1 - 0.3158 - 0.1108) = 0.5564$$

$$E_p(4) = 0.9704 \times \frac{1}{1 - 1/\infty} \times (1 - 0.3158 - 0.1108) = 0.5564$$

$$E_p(4) = 0.9704 \times (1 - 0.3158 - 0.1108) = 0.5564$$

$$E_p(4) = 0.9704 \times \frac{1}{1.000} \times \frac{70 + 50 + 48 + 39}{361} = 0.5564$$

$$E_p(4) = ((70 + 50 + 48 + 39) + 0) / (93 \times 4) = 0.5565$$

【0 1 7 9】

以上計算された並列性能評価指標をまとめると図 4 5 のようになる。 $A_p(4) = \infty$ ゆえ $p = \infty$ で並列処理した時の性能向上の可能性は無限であるが、 $p = 4$ を投入した現実の性能向上 $E_p \cdot p$ は 2.226 である。その理由は、並列効率 E_p が通信により 32% ($R_C(4) = 0.3158$)、アイドルングにより 11% ($R_W(4) = 0.1108$) 低下するためである。ロードバランス寄与率は 97% ($R_b(4) = 0.97043$) で、ロードバランスはこの場合並列性能を阻害する主要な要因ではない。

【0 1 8 0】

(5) ホモ構成におけるグリッド等（他の処理があるために待ちがある場合）

グリッドやクラスタで処理を行う場合、各プロセッサを自分の処理のみで使うことは希で、一般に複数の処理の中に共存することになる。その場合他の処理が割り込むことによる待ちが生じる。これを図 4 6 に示す。このように他の処理があるために待ちがある場合の並列性能を評価する。図 4 6 のような経過時間の測定結果が得られたものとする。

【0 1 8 1】

式 (3) から以下の計算がなされる。

【数 9 6】

$$\tau_1(4) = 5 + 25 + 50 + 35 + 5 = 120$$

$$\tau_2(4) = 10 + 20 + 5 + 10 + 10 + 30 + 4 = 89$$

$$\tau_3(4) = 15 + 15 + 10 + 10 + 10 + 33 + 3 = 96$$

$$\tau_4(4) = 10 + 10 + 20 + 29 + 4 = 73$$

式 (5)、式 (12-1)、式 (6-1)、式 (6-2) 及び式 (7) からそれぞれ以下のような計算がなされる。

【数 9 7】

$$R_b(4) = \frac{120 + 89 + 96 + 73}{120 \times 4} = 0.7875$$

$$R_p(4) = \frac{60 + 60 + 58 + 39}{60 + 60 + 58 + 39} = 1.000$$

$$A_p(4) = \frac{1}{1-1} = \infty$$

$$R_C(4) = \frac{10 + 14 + 18 + 14}{378} = 0.1481$$

$$R_W(4) = \frac{50 + 15 + 20 + 20}{378} = 0.2778$$

並列効率については、順番に式 (4-4)、式 (4-5)、式 (8-2)、式 (9-1)、式 (9-2) からそれぞれ以下のような計算がなされる。

【数 9 8】

$$E_p(4) = 0.7875 \times \frac{1}{1.000} \times (1 - 0.1481 - 0.2778) = 0.4521$$

$$E_p(4) = 0.7875 \times \frac{1}{1 - 1/\infty} \times (1 - 0.1418 - 0.2778) = 0.4521$$

$$E_p(4) = 0.7875 \times (1 - 0.1418 - 0.2778) = 0.4521$$

$$E_p(4) = 0.7875 \times \frac{1}{1.000} \times \frac{60 + 60 + 58 + 39}{378} = 0.4521$$

$$E_p(4) = ((60 + 60 + 58 + 39) + 0) / (120 \times 4) = 0.4521$$

【0 1 8 2】

以上計算された並列性能評価指標をまとめると図 4 7 のようになる。 $A_p(4) = \infty$ ゆえ $p = \infty$ で並列処理した時の性能向上の可能性は無限であるが、 $p = 4$ を投入した現実の性能向上 $E_p \cdot p$ は 1.808 である。その理由は、並列効率 E_p が、ロードバランス寄与率で 79% ($R_b(4) = 0.7875$) となり、タイムシェアリングのための待ちにより 28% ($R_w(4) = 0.2778$)、通信によりさらに 14% ($R_c = 0.1418$) 低下するためである。 $R_w(4)$ は他の処理により生じるので、 $R_b(4)$ はシステム全体を考慮したロードバランス寄与率となる。他の処理がある場合、 $R_b(4)$ と $R_w(4)$ に注目する必要がある。たとえ $R_b(4) = 1$ であっても、 $R_w(4)$ が大きければそれは混んだシステムを利用していることになり、 E_p は低い値となる。グリッド又はクラスタ処理を展開する際、特に $R_b(4)$ が 1 に近づくように且つ R_w が 0 になるようにシステムを選択することで、並列処理を効率良く行うことが可能となる。このような事が分かるのは本実施の形態が初めてである。

【0 1 8 3】

尚、自処理（目的の処理）か他処理（目的外の処理）かを見分ける方法として、CPU 時間と経過時間を測定する方法がある。一般に CPU 時間は自処理のみの時間、経過時間は他処理を含んだ時間となる。従ってタイムシェアリングのた

めの待ち時間＝経過時間－CPU時間とできる場合がある。

【0184】

(6) ホモ構成におけるグリッド等（データパラレル処理の場合）

データパラレル処理は、例えば1000件のデータを4プロセッサで250件ずつ分割して処理するような、各プロセッサの手続きが同じでデータが異なる並列処理である。並列処理できない処理は、全プロセッサのデータを同じにする、すなわち冗長処理を行う場合と、あるプロセッサで処理して全プロセッサに放送する場合がある。ここでは両者の並列性能を評価する。

【0185】

[冗長処理を用いたデータパラレル処理]

図48のような経過時間の測定結果が得られたものとする。また、 $x_{1,C}=0$ とする。

【0186】

式(3)、式(5)、式(12-1)、式(6-1)、式(6-2)及び式(7)からそれぞれ以下のような計算がなされる。

【数 9 9】

$$\tau_1(4) = \tau_2(4) = \tau_3(4) = \tau_4(4) = 5 + 20 + 5 + 10 + 2 + 30 + 3 = 75$$

$$R_b(4) = \frac{75 + 75 + 75 + 75}{75 \times 4} = 1.0000$$

$$\chi_{1,RED}(1) = \frac{1}{4} \cdot (10 + 10 + 10 + 10) = 10.00$$

$$R_p(4) = \frac{50 + 50 + 50 + 50}{(50 + 50 + 50 + 50) + 10} = 0.9524$$

$$A_p(4) = \frac{1}{1 - 0.9524} = 21.01$$

$$R_{RED}(4) = \frac{10 + 10 + 10 + 10}{300} = 0.1333$$

$$R_C(4) = \frac{15 + 15 + 15 + 15}{300} = 0.2000$$

並列効率については、順番に式（４－４）、式（４－５）、式（９－１）、式（９－２）からそれぞれ以下のような計算がなされる。

【数 1 0 0】

$$E_p(4) = 1.000 \times \frac{1}{0.9524} \times (1 - 0.1333 - 0.2000) = 0.7000$$

$$E_p(4) = 1.000 \times \frac{1}{1 - 1/21.01} \times (1 - 0.1333 - 0.2000) = 0.7000$$

$$E_p(4) = 1.000 \times \frac{1}{0.9524} \times \frac{50 + 50 + 50 + 50}{75 + 75 + 75 + 75} = 0.7000$$

$$E_p(4) = ((50 + 50 + 50 + 50) + 10) / (75 \times 4) = 0.7000$$

【 0 1 8 7】

以上計算された並列性能評価指標をまとめると図 4 9 のようになる。 $A_p(4) =$

2.1 ゆえ、プロセッサ数は $p \leq 2.1$ で選択すべきである。プロセッサ数 $p = 4$ を投入した時の現実の性能向上 $E_p \cdot p$ は 2.800 である。その理由は、並列効率 $E_p(4)$ が通信で 20% ($R_C = 0.2000$)、冗長処理で 13% ($R_{RED}(4) = 0.1333$) 低下するためである。

【0188】

[並列処理できない部分を特定のプロセッサで処理するデータパラレル処理]

並列処理できない部分を冗長処理する代わりに、特定のプロセッサで処理する場合がある。図 50 は図 48 の冗長処理の代わりにプロセッサ # 1 でのみ処理を行い (γ' の部分)、結果を各プロセッサに放送した場合である。当然その間、他のプロセッサはプロセッサ # 1 の結果待ちとなる。またここでは γ' を並列処理として取り扱ったが、逐次処理として並列処理阻害要因に加えれば、より詳細な並列性能評価ができる。しかしそのためには γ' の処理が逐次処理か並列処理かの判別が必要となる。図 50 のような経過時間の測定結果が得られたものとする。また、 $x_{1,C} = 0$ とする。

【0189】

式 (3)、式 (5)、式 (12-1)、式 (6-1)、式 (6-2) 及び式 (7) からそれぞれ以下のような計算がなされる。

【数 1 0 1】

$$\tau_1(4) = \tau_2(4) = \tau_3(4) = \tau_4(4) = 5 + 20 + 5 + 10 + 2 + 30 + 3 = 75$$

$$R_b(4) = \frac{75 + 75 + 75 + 75}{75 \times 4} = 1.0000$$

$$R_p(4) = \frac{50 + 50 + 50 + 50}{(50 + 50 + 50 + 50) + 0} = 1.000$$

$$A_p(4) = \frac{1}{1-1} = \infty$$

$$R_C(4) = \frac{15 + 15 + 15 + 15}{300} = 0.2000$$

$$R_W(4) = \frac{0 + 10 + 10 + 10}{300} = 0.1000$$

並列効率については、順番に式（４－４）、式（４－５）、式（９－１）、式（９－２）からそれぞれ以下のような計算がなされる。

【数 1 0 2】

$$E_p(4) = 1.000 \times \frac{1}{1.000} \times (1 - 0.1000 - 0.2000) = 0.7000$$

$$E_p(4) = 1.000 \times \frac{1}{1-1/\infty} \times (1 - 0.1000 - 0.2000) = 0.7000$$

$$E_p(4) = 1.000 \times \frac{1}{1.000} \times \frac{60 + 50 + 50 + 50}{75 + 75 + 75 + 75} = 0.7000$$

$$E_p(4) = ((60 + 50 + 50 + 50) + 0) / (75 \times 4) = 0.7000$$

【0 1 9 0】

以上計算された並列性能評価指標をまとめると図 5 1 のようになる。 $A_p(4) = \infty$ で並列処理した時の性能向上は無限であるが、 $p = 4$ を投入した時の現実の性

能向上 $E_p \cdot p$ は 2.800 である。その理由は、並列効率 $E_p(4)$ が通信で 20% ($R_c(4) = 0.2000$)、待ちで 10% ($R_w(4) = 0.1000$) 低下するためである。図 4 9 と図 5 1 では $R_p(4)$ 、 $A_p(4)$ 、 $R_{RED}(4)$ 、 R_w の値が異なる。一方 $R_b(4)$ 及び $E_p(4)$ は同じ値となる。図 5 1 では並列処理できない部分 γ' を並列処理として評価したため、 $R_p(4) = 1$ となった。またプロセッサ # 2, 3, 4 の冗長処理が待ちに変わり、並列性能阻害要因 $R_w(4)$ に代替される。

【 0 1 9 1 】

(7) ホモ構成におけるグリッド等（コントロールパラレル処理の場合）

コントロールパラレル処理は、通常各プロセッサの手続きが異なる。このため各プロセッサの手続き時間がばらばらな並列処理となる場合が多い。ここではコントロールパラレルの並列性能を評価する。図 5 2 のような経過時間の測定結果が得られたものとする。また、 $x_{1,C} = 0$ とする。

【 0 1 9 2 】

式 (3) から以下の計算がなされる。

【数 1 0 3】

$$\tau_1(4) = 53 + 15 + 3 + 12 + 3 + 20 = 106$$

$$\tau_2(4) = 2 + 30 + 5 + 10 + 4 + 2 + 15 + 3 = 71$$

$$\tau_3(4) = 4 + 20 + 7 + 20 + 2 + 30 + 3 = 86$$

$$\tau_4(4) = 6 + 20 + 5 + 20 + 2 + 30 + 3 = 86$$

式 (5)、式 (6-1)、式 (6-2) 及び式 (7) からそれぞれ以下のような計算がなされる。

【数 1 0 4】

$$R_b(4) = \frac{106 + 71 + 86 + 86}{106 \times 4} = 0.8231$$

$$R_p(4) = \frac{73 + 55 + 70 + 70}{73 + 55 + 70 + 70} = 1.000$$

$$A_p(4) = \frac{1}{1 - 1.000} = \infty$$

$$R_{TC}(4) = \frac{0 + 2 + 4 + 6}{349} = 0.0344$$

$$R_C(4) = \frac{6 + 10 + 12 + 10}{349} = 0.1089$$

$$R_W(4) = \frac{27 + 4 + 0 + 0}{349} = 0.0888$$

並列効率については、順番に式（4－4）、式（4－5）、式（8－2）、式（9－1）、式（9－2）からそれぞれ以下のような計算がなされる。

【数 1 0 5】

$$E_p(4) = 0.8231 \times \frac{1}{1.000} \times (1 - 0.0344 - 0.1089 - 0.0888) = 0.6321$$

$$E_p(4) = 0.8231 \times \frac{1}{1 - 1/\infty} \times (1 - 0.0344 - 0.1089 - 0.0888) = 0.6321$$

$$E_p(4) = 0.8231 \times (1 - 0.0344 - 0.1089 - 0.0888) = 0.6321$$

$$E_p(4) = 0.8231 \times \frac{1}{1.000} \times \frac{73 + 55 + 70 + 70}{106 + 71 + 86 + 86} = 0.6321$$

$$E_p(4) = ((73 + 55 + 70 + 70) + 0) / (106 \times 4) = 0.6321$$

【0 1 9 3】

以上計算された並列性能評価指標をまとめると図 5 3 のようになる。 $A_p(4) = \infty$ で並列処理した時の性能向上は無限であるが、 $p = 4$ を投入した時の現実の性能向上 $E_p \cdot p$ は 2.528 である。その理由は、並列効率 E_p がロードバランス寄与率で 82% ($R_b(4) = 0.8231$)、さらにタスク生成、通信、待ちをあわせて 23% ($R_{TC}(4) + R_C(4) + R_W(4) = 0.0344 + 0.1089 + 0.0888$) 低下するためである。

【 0 1 9 4 】

並列性能の向上を図るには、並列性能指標を比べ、並列性能の低下に影響力の大きい順に改善の余地を検討する。図 5 3 の場合、これは $R_b(4)$ 、 $R_C(4)$ 、 $R_W(4)$ 、 $R_{TC}(4)$ の順となる。 $R_b(4) = 1$ になれば $E_p(4) \cdot p = 3.071 (= 2.528 / 0.8231)$ となる。そのため例えば、プロセッサ # 1 の処理時間が他のプロセッサと同じになるように処理スケジュールを変更することを試みる。次の改善の余地は $R_C(4)$ の削減である。削減方法としては例えば通信性能が 2 倍になるようなハードウェアに置きかえることが考えられる。その場合には、以下のような計算がなされる。

まず、式 (3) から以下の計算がなされる。

【数 1 0 6】

$$\tau_1(4) = 53 + 15 + 3/2 + 12 + 3/2 + 20 = 103$$

$$\tau_2(4) = 2 + 30 + 5/2 + 10 + 4 + 2/2 + 15 + 3/2 = 66$$

$$\tau_3(4) = 4 + 20 + 7/2 + 20 + 2/2 + 30 + 3/2 = 80$$

$$\tau_4(4) = 6 + 20 + 5/2 + 20 + 2/2 + 30 + 3/2 = 81$$

式 (5)、式 (6-1)、式 (6-2) 及び式 (7) からそれぞれ以下のような計算がなされる。

【数 1 0 7】

$$R_b(4) = \frac{103 + 66 + 80 + 81}{103 \times 4} = 0.8010$$

$$R_p(4) = \frac{73 + 55 + 70 + 70}{73 + 55 + 70 + 70} = 1.000$$

$$A_p(4) = \frac{1}{1 - 1.000} = \infty$$

$$R_{TC}(4) = \frac{0 + 2 + 4 + 6}{330} = 0.0364$$

$$R_C(4) = \frac{(6 + 10 + 12 + 10) / 2}{330} = 0.0576$$

$$R_W(4) = \frac{27 + 4 + 0 + 0}{330} = 0.0939$$

並列効率については、式（4-4）以下のような計算がなされる。

【数 1 0 8】

$$E_p(4) = 0.8010 \times \frac{1}{1.000} \times (1 - 0.0364 - 0.0576 - 0.0939) = 0.6505$$

さらに、以下のような計算もなされる。

【数 1 0 9】

$$E_p(4) \cdot p = 0.8010 \times \frac{1}{1.000} \times (1 - 0.0364 - 0.0576 - 0.0939) \cdot 4 = 0.6505 \cdot 4 = 2.602$$

もし、通信の性能を上記のように向上させ、さらに $R_b(4) = 1$ にロードバランスを変更できれば、以下のような計算がなされる。

【数 1 1 0】

$$E_p(4) \cdot p = 1 \times \frac{1}{1.000} \times (1 - 0.0364 - 0.0576 - 0.0939) \cdot 4 = 0.8121 \cdot 4 = 3.248$$

【 0 1 9 5】

E. チューニング処理において示したように、本実施の形態では、各並列性能阻害要因をチューニングして改善した時の並列性能を推測できる。従来のチューニングでは目標値を処理時間にしていたため、不可能な目標値が設定されることがあったが、本実施の形態では E_p を用いてリーズナブルな目標設定が可能となる。さらに本実施の形態では、並列効率等を 1 回の測定結果で計算することができ、チューニング時の性能評価時間を短縮することが可能である。さらに、従来のチューニングでは、入力データや処理機能を変更するとそれまで測定した各並列性能阻害要因に対する処理時間を性能評価に使うことができなくなる。従って入力データや処理機能毎に独立した並列性能評価を行ってきた。本実施の形態では性能評価指標がすべて比率の形になっており、異なった入力データや処理機能の並列性能を比較できる。

【0 1 9 6】

(8) ホモ構成におけるグリッド等（コントロールパラレルでマスタ・スレイブ処理を行う場合）

コントロールパラレル処理は、通常各プロセッサの手続きが異なる。マスタ・スレイブ処理の場合、1 つのプロセッサが他のプロセッサの管理をするマスタとなり、その指示に従って複数のプロセッサが処理を実施する。ここではプロセッサ # 1 をマスタプロセッサとした場合の並列性能を評価する。図 5 4 のような経過時間の測定結果が得られたものとする。また、 $\alpha_{1,C} = 0$ とする。

【0 1 9 7】

式 (3) から以下の計算がなされる。

【数 1 1 1】

$$\tau_1(4) = 2 + 12 + 2 + 5 + 2 + 3 + 2 + 5 + 2 + 5 + 2 + 43 + 6 = 91$$

$$\tau_2(4) = 2 + 2 + 10 + 2 + 5 + 2 + 17 + 2 = 42$$

$$\tau_3(4) = 4 + 2 + 20 + 2 + 5 + 2 + 50 + 3 = 88$$

$$\tau_4(4) = 6 + 2 + 80 + 3 = 91$$

式 (5)、式 (6-1)、式 (6-2) 及び式 (7) からそれぞれ以下のような

計算がなされる。

【数 1 1 2】

$$R_b(4) = \frac{91 + 42 + 88 + 91}{91 \times 4} = 0.8571$$

$$R_p(4) = \frac{10 + 27 + 70 + 80}{10 + 27 + 70 + 80} = 1.000$$

$$A_p(4) = \frac{1}{1 - 1.000} = \infty$$

$$R_{TC}(4) = \frac{0 + 2 + 4 + 6}{312} = 0.0385$$

$$R_C(4) = \frac{18 + 8 + 9 + 5}{312} = 0.1282$$

$$R_W(4) = \frac{63 + 5 + 5 + 0}{312} = 0.2340$$

並列効率については、順番に式（４－４）、式（４－５）、式（８－２）、式（９－１）、式（９－２）からそれぞれ以下のような計算がなされる。

【数 1 1 3】

$$E_p(4) = 0.8571 \times \frac{1}{1.000} \times (1 - 0.0385 - 0.1282 - 0.2340) = 0.5137$$

$$E_p(4) = 0.8571 \times \frac{1}{1 - 1/\infty} \times (1 - 0.0385 - 0.1282 - 0.2340) = 0.5137$$

$$E_p(4) = 0.8571 \times (1 - 0.0385 - 0.1282 - 0.2240) = 0.5137$$

$$E_p(4) = 0.8571 \times \frac{1}{1.000} \times \frac{10 + 27 + 70 + 80}{312} = 0.5137$$

$$E_p(4) = (10 + 27 + 70 + 80 + 0) / (91 \times 4) = 0.5137$$

【 0 1 9 8 】

以上計算された並列性能評価指標をまとめると図 5 5 のようになる。 $A_p(4) = \infty$ ゆえ $p = \infty$ で並列処理した時の性能向上は無限であるが、 $p = 4$ を投入した時の現実の性能向上 $E_p \cdot p$ は 2.055 である。その理由は、並列効率 E_p がロードバランス寄与率で 86% ($R_b(4) = 0.8571$)、さらに待ちで 23% ($R_w(4) = 0.2340$)、タスク生成、通信をあわせて 17% ($R_{TC}(4) + R_C(4) = 0.0385 + 0.1282$) 低下するためである。マスタスレイブ処理を行う場合、マスタプロセッサの待ち時間が処理全体の性能に重要な影響を及ぼすことが知られているが、本実施の形態では待ちが性能に及ぼす影響を定量的に捉え、マスタスレイブ処理が有効に行われているかを判断することができる。

【 0 1 9 9 】

(9) ホモ構成におけるグリッド等（データパラレルとコントロールパラレル混在の場合）

データパラレルとコントロールパラレルを混在させた処理は、ロードバランスを保たせることが難しいため通常の業務では使用されない。本実施の形態ではこのような場合の並列性能評価も可能となる。本実施の形態は、処理のコントロールのための性能評価指標を提供するため、このような処理に対する実用的な評価方法を提供するものである。ここではプロセッサ # 1 乃至 # 4 はコントロールパラレルで、プロセッサ # 5 乃至 # 8 はデータパラレルで、プロセッサ # 1 をマスタプロセッサとした場合の並列性能を評価する。図 5 6 のような経過時間の測定結果が得られたものとする。また、 $x_{1,C} = 0$ とする。

【 0 2 0 0 】

式 (3) から以下の計算がなされる。

【数 1 1 4】

$$\tau_1(8) = 2 + 12 + 2 + 5 + 2 + 3 + 2 + 5 + 2 + 5 + 2 + 43 + 6 = 91$$

$$\tau_2(8) = 2 + 2 + 10 + 2 + 5 + 2 + 17 + 2 = 42$$

$$\tau_3(8) = 4 + 2 + 20 + 2 + 5 + 2 + 50 + 3 = 88$$

$$\tau_4(8) = 6 + 2 + 80 + 3 = 91$$

$$\tau_5(8) = \tau_6(8) = \tau_7(8) = \tau_8(8) = 2 + 2 + 30 + 3 + 10 + 2 + 40 + 2 = 91$$

式 (5)、式 (12-1)、式 (6-1)、式 (6-2) 及び式 (7) からそれぞれ以下のような計算がなされる。

【数 1 1 5】

$$R_b(8) = \frac{91 + 42 + 88 + 91 + 91 + 91 + 91 + 91}{91 \times 8} = \frac{676}{728} = 0.9286$$

$$\chi_{1,RED}(1) = \frac{1}{4} \cdot (10 + 10 + 10 + 10) = 10$$

$$R_p(8) = \frac{10 + 27 + 70 + 80 + 70 + 70 + 70 + 70}{(10 + 27 + 70 + 80 + 70 + 70 + 70 + 70) + 10} = \frac{467}{477} = 0.9790$$

$$A_p(8) = \frac{1}{1 - 0.9790} = 47.62$$

$$R_{RED}(8) = \frac{0 + 0 + 0 + 0 + 10 + 10 + 10 + 10}{676} = 0.0592$$

$$R_{TC}(8) = \frac{0 + 2 + 4 + 6 + 2 + 2 + 2 + 2}{676} = 0.0296$$

$$R_C(8) = \frac{18 + 8 + 9 + 5 + 9 + 9 + 9 + 9}{676} = 0.1124$$

$$R_W(8) = \frac{63 + 5 + 5 + 0 + 0 + 0 + 0 + 0}{676} = 0.1080$$

並列効率については、順番に式 (4-4)、式 (4-5)、式 (8-2)、式 (9-1)、式 (9-2) からそれぞれ以下のような計算がなされる。

【数 1 1 6】

$$E_p(8) = 0.9286 \times \frac{1}{0.9790} \times (1 - 0.0592 - 0.0296 - 0.1124 - 0.1080) = 0.6552$$

$$E_p(8) = 0.9286 \times \frac{1}{1 - 1/47.62} \times (1 - 0.0592 - 0.0296 - 0.1124 - 0.1080) = 0.6552$$

$$E_p(8) = 0.9286 \times (1 - 0.0592 - 0.0296 - 0.1124 - 0.1080) = 0.6552$$

$$E_p(8) = 0.9286 \times \frac{1}{0.9790} \times \frac{467}{676} = 0.6553$$

$$E_p(8) = (467 + 10) / (91 \times 8) = 0.6552$$

【0 2 0 1】

以上計算された並列性能評価指標をまとめると図 5 7 のようになる。 $A_p(8) = 47.62$ ゆえ、 $p < 47$ で並列処理すべきである。プロセッサ数 $p = 8$ を投入した時の現実の性能向上 $E_p \cdot p$ は 5.242 である。その理由は、並列効率 E_p がロードバランス寄与率で 93% ($R_b(8) = 0.9286$)、さらに待ちで 11% ($R_w(8) = 0.1080$)、通信で 11% ($R_c(8) = 0.1124$)、冗長処理、タスク生成をあわせて 9% ($R_{RED}(8) + R_{TC}(8) = 0.0592 + 0.0296$) 低下するためである。このように本実施の形態はデータパラレルとコントロールパラレル混在という並列処理方式に適用することができる。

【0 2 0 2】

(10) グリッド等のヘテロ構成で冗長処理がある場合 ($x_{1,RED} \neq 0$)

グリッドやクラスタでつながれたプロセッサは、CPU の能力が異なっている場合が多い。これをヘテロ構成と呼ぶ。本実施の形態では、ヘテロ構成の場合にも適用できる。ここでは実施例 (2) においてプロセッサ # 1 が 1/2 の性能である場合の並列性能を評価する。図 5 8 のような経過時間の測定結果が得られたものとする。

【0 2 0 3】

式 (3) から以下の計算がなされる。

【数 1 1 7】

$$\tau_1(4) = 16 + 70 + 20 + 40 + 10 = 156$$

$$\tau_2(4) = 10 + 33 + 11 + 22 + 7 = 83$$

$$\tau_3(4) = 7 + 37 + 10 + 19 + 4 = 77$$

$$\tau_4(4) = 11 + 30 + 9 + 18 + 6 = 74$$

式 (5)、式 (12-1)、式 (6-1)、式 (6-2) 及び式 (7) からそれぞれ以下のような計算がなされる。

【数 1 1 8】

$$R_b(4) = \frac{156 + 83 + 77 + 74}{156 \times 4} = \frac{390}{624} = 0.6250$$

$$\chi_{1,RED}(1) = \frac{1}{4} \cdot (46 + 28 + 21 + 26) = 30.25$$

$$R_p(4) = \frac{110 + 55 + 56 + 48}{(110 + 55 + 56 + 48) + 30.25} = \frac{269}{299.3} = 0.8988$$

$$A_p(4) = \frac{1}{1 - 0.8988} = 9.881$$

$$R_{RED}(4) = \frac{46 + 28 + 21 + 26}{156 + 83 + 77 + 74} = \frac{121}{390} = 0.3103$$

並列効率については、順番に式 (4-4)、式 (4-5)、式 (9-1)、式 (9-2) からそれぞれ以下のような計算がなされる。

【数 1 1 9】

$$E_p(4) = 0.6250 \times \frac{1}{0.8988} \times (1 - 0.3103) = 0.4796$$

$$E_p(4) = 0.6250 \times \frac{1}{1 - 1/9.881} \times (1 - 0.3103) = 0.4796$$

$$E_p(4) = 0.6250 \times \frac{1}{0.8988} \times \frac{269}{390} = 0.4796$$

$$E_p(4) = (269 + 30.25) / (156 \times 4) = 0.4796$$

【0 2 0 4】

以上計算された並列性能評価指標をまとめると図 5 9 のようになる。 $A_p(4) = 9.881$ ゆえ、 $p > 9$ の並列処理は無意味である。 $p = 4$ のプロセッサを投入した現実の性能向上 $E_p \cdot p$ は1.918である。その理由は、並列効率 E_p が、ロードバランス寄与率で63% ($Rb(4) = 0.6250$)となり、冗長処理によりさらに31% ($R_{RED}(4) = 0.3103$)低下するためである。図 4 1 と比較するとロードバランス寄与率 $Rb(4)$ が0.9398から0.6250に低下することが分かる。これは図 4 1 と図 5 9 に示されるようにプロセッサ # 1 の違いが性能評価指標 $Rb(4)$ に反映された結果である。一般に等分割したタスクをCPU能力が異なったプロセッサで処理するとロードバランスが崩れる。本実施の形態ではこれを $Rb(4)$ によって検知することができる。

【0 2 0 5】

(付記 1)

並列計算機システムの並列効率を計算する並列効率計算方法であって、

前記並列計算機システムに含まれる各プロセッサ間の負荷の均衡度合いを表すロードバランス寄与率を計算し、記憶装置に格納するロードバランス寄与率計算ステップと、

前記並列計算機システムにおいて実施した処理のうち各プロセッサにより並列計算された部分の、時間についての割合を表す仮想並列化率を計算し、記憶装置

に格納する仮想並列化率計算ステップと、

前記並列計算機システムに含まれる全プロセッサの全処理時間に対する各並列性能阻害要因部分の処理時間の割合を表す並列性能阻害要因寄与率を計算し、記憶装置に格納する並列性能阻害要因寄与率計算ステップと、

前記ロードバランス寄与率と前記仮想並列化率と前記並列性能阻害要因寄与率とを用いて並列効率を計算し、記憶装置に格納するステップと、

を含む並列効率計算方法。

【 0 2 0 6 】

(付記 2)

並列計算機システムの並列効率を計算する並列効率計算方法であって、

前記並列計算機システムに含まれる各プロセッサ間の負荷の均衡度合いを表すロードバランス寄与率を計算し、記憶装置に格納するロードバランス寄与率計算ステップと、

前記並列計算機システムにおいて実施する処理の並列化による処理時間の短縮度合いの向上の限度を表す加速率を計算し、記憶装置に格納する加速率計算ステップと、

前記並列計算機システムに含まれる全プロセッサの全処理時間に対する各並列性能阻害要因部分の処理時間の割合を表す並列性能阻害要因寄与率を計算し、記憶装置に格納する並列性能阻害要因寄与率計算ステップと、

前記ロードバランス寄与率と前記加速率と前記並列性能阻害要因寄与率とを用いて並列効率を計算し、記憶装置に格納するステップと、

を含む並列効率計算方法。

【 0 2 0 7 】

(付記 3)

並列計算機システムの並列効率を計算する並列効率計算方法であって、

前記並列計算機システムに含まれる各プロセッサ間の負荷の均衡度合いを表すロードバランス寄与率を計算し、記憶装置に格納するロードバランス寄与率計算ステップと、

前記並列計算機システムに含まれる全プロセッサの全処理時間に対する各並列

性能阻害要因部分の処理時間の割合を表す並列性能阻害要因寄与率を計算し、記憶装置に格納する並列性能阻害要因寄与率計算ステップと、

前記ロードバランス寄与率と前記並列性能阻害要因寄与率とを用いて並列効率を計算し、記憶装置に格納するステップと、
を含む並列効率計算方法。

【 0 2 0 8 】

(付記 4)

並列計算機システムの並列効率を計算する並列効率計算方法であって、

前記並列計算機システムに含まれる各プロセッサ間の負荷の均衡度合いを表すロードバランス寄与率を計算し、記憶装置に格納するロードバランス寄与率計算ステップと、

前記並列計算機システムにおいて実施する処理のうち各プロセッサにより並列計算される部分の、時間についての割合を表す仮想並列化率を計算し、記憶装置に格納する仮想並列化率計算ステップと、

前記並列計算機システムに含まれる各プロセッサにおいて実施された処理のうち並列計算部分の処理時間の和と、前記各プロセッサにおいて実施された処理の処理時間の和と、前記ロードバランス寄与率と、前記仮想並列化率とを用いて並列効率を計算し、記憶装置に格納するステップと、

を含む並列効率計算方法。

【 0 2 0 9 】

(付記 5)

並列計算機システムの並列効率を計算する並列効率計算方法であって、

1 プロセッサにより処理を実施する場合において当該処理のうち並列性能阻害部分の全処理時間に相当する第 1 の処理時間を計算し、記憶装置に格納するステップと、

前記並列計算機システムに含まれる各プロセッサにおいて実施された処理のうち並列計算部分の処理時間の和である第 2 の処理時間を計算し、記憶装置に格納するステップと、

前記並列計算機システムにおいて使用したプロセッサの数と、前記並列計算機

システムに含まれる各プロセッサにおいて実施された処理の処理時間のうち最長の処理時間と、前記第 1 の処理時間と、前記第 2 の処理時間とを用いて並列効率を計算し、記憶装置に格納するステップと、

を含む並列効率計算方法。

【 0 2 1 0 】

(付記 6)

前記ロードバランス寄与率計算ステップにおいて、

前記ロードバランス寄与率を、

前記並列計算機システムに含まれる全プロセッサにおいて実施された処理の全処理時間を、前記並列計算機システムに含まれる各プロセッサにおいて実施された処理の処理時間のうち最長の処理時間及び前記並列計算機システムにおいて使用したプロセッサ数により除することにより計算する

ことを特徴とする付記 1 乃至 4 のいずれか 1 つ記載の並列効率計算方法。

【 0 2 1 1 】

(付記 7)

前記仮想並列化率計算ステップにおいて、

前記仮想並列化率を、

前記並列計算機システムに含まれる各プロセッサにおいて実施された処理のうち並列計算部分の処理時間の和を、1 プロセッサにより同一処理を実施した場合の第 3 の処理時間に相当する処理時間により除することにより計算する

ことを特徴とする付記 1 又は 4 記載の並列効率計算方法。

【 0 2 1 2 】

(付記 8)

前記並列性能阻害要因寄与率計算ステップにおいて、

特定の並列性能阻害要因についての並列性能阻害要因寄与率を、

前記並列計算機システムに含まれる各プロセッサにおける前記特定の並列性能阻害要因部分の処理時間の和を、前記並列計算機システムに含まれる各プロセッサの処理時間の和により除することにより計算する

ことを特徴とする付記 1 乃至 3 のいずれか 1 つ記載の並列効率計算方法。

【 0 2 1 3 】

(付記 9)

前記加速率計算ステップにおいて、

前記加速率を、

前記並列計算機システムに含まれる各プロセッサにおいて実施された処理のうち並列計算部分の処理時間の和を 1 プロセッサにより同一処理を実施した場合の第 3 の処理時間に相当する処理時間により除することにより計算される仮想並列化率を 1 から差し引いた値の逆数として計算する

ことを特徴とする付記 2 記載の並列効率計算方法。

【 0 2 1 4 】

(付記 1 0)

前記処理時間が、対応する事象の確認回数で表されることを特徴とする付記 1 乃至 9 のいずれか 1 つ記載の並列効率計算方法。

【 0 2 1 5 】

(付記 1 1)

計算された前記並列効率に前記並列計算機システムにおいて使用したプロセッサ数を乗じて補助指標を計算し、記憶装置に格納するステップ、

をさらに含む付記 1 乃至 1 0 のいずれか 1 つ記載の並列効率計算方法。

【 0 2 1 6 】

(付記 1 2)

前記第 3 の処理時間を、

1 プロセッサにより処理を実施する場合において当該処理のうち並列性能阻害部分の全処理時間に相当する第 1 の処理時間と前記並列計算機システムに含まれる各プロセッサにおいて実施された処理のうち並列計算部分の処理時間の和である第 2 の処理時間との和により計算する

ことを特徴とする付記 7 又は 9 のいずれか 1 つ記載の並列効率計算方法。

【 0 2 1 7 】

(付記 1 3)

前記第 1 の処理時間が、前記並列計算機システムに含まれる各プロセッサにお

いて実施された処理のうち冗長処理の処理時間の和をプロセッサ数で除した値、前記並列計算機システムに含まれる各プロセッサにおいて実施された処理のうち冗長処理の処理時間の最大値、前記並列計算機システムに含まれる各プロセッサにおいて実施された処理のうち冗長処理の処理時間の最小値、前記並列計算機システムに含まれる各プロセッサにおいて実施された処理のうち並列処理の処理時間と並列性能阻害要因の処理時間の総和が最大となるプロセッサにおける冗長処理の処理時間の値のいずれかである

ことを特徴とする付記 5 又は 1 2 記載の並列効率計算方法。

【 0 2 1 8 】

(付記 1 4)

前記第 1 の処理時間が、前記並列計算機システムに含まれる各プロセッサにおいて実施された処理のうち冗長処理以外の並列性能阻害要因による処理時間から 2 以上のプロセッサ数で発生し且つプロセッサ数に依存する並列化阻害要因による処理時間を減じた第 4 の処理時間を全プロセッサについて加算した値をプロセッサ数で除した値、全プロセッサにおける前記第 4 の処理時間の最大値、全プロセッサにおける前記第 4 の処理時間の最小値のいずれかの値である

ことを特徴とする付記 5 又は 1 2 記載の並列効率計算方法。

【 0 2 1 9 】

(付記 1 5)

目標並列効率を設定するステップと、

計算された前記並列効率とプロセッサ数の積を前記目標並列効率で除することにより最適プロセッサ数を計算し、記憶装置に格納するステップと、

をさらに含む付記 1 乃至 1 4 のいずれか 1 つ記載の並列効率計算方法。

【 0 2 2 0 】

(付記 1 6)

システム増強時における増加分の稼働時間と予測並列効率とを設定するステップと、

前記並列計算機システムに現在含まれる各プロセッサにおいて実施された処理の処理時間の和と計算された前記並列効率との全処理についての積和と、前記増

加分の稼働時間及び前記予測並列効率の積との和を、前記並列計算機システムに現在含まれる各プロセッサの稼働時間の和で除することにより、システム増強時の加速率を計算し、記憶装置に格納するステップと、

をさらに含む付記 1 乃至 1 4 のいずれか 1 つ記載の並列効率計算方法。

【 0 2 2 1 】

(付記 1 7)

前記並列計算機システムに対する新たな並列計算機システムの性能倍率を設定するステップと、

前記新たな並列計算機システムの性能倍率を用いて見積並列効率を計算し、記憶装置に格納するステップと、

をさらに含む付記 1 乃至 1 4 のいずれか 1 つ記載の並列効率計算方法。

【 0 2 2 2 】

(付記 1 8)

前記並列計算機システムに現在含まれる各プロセッサにおいて実施された処理の処理時間の和と計算された前記並列効率との全処理についての積和を、前記並列計算機システムに現在含まれる各プロセッサの全稼働時間で除することにより、システム運用効率を計算し、記憶装置に格納するステップと、

をさらに含む付記 1 乃至 1 4 のいずれか 1 つ記載の並列効率計算方法。

【 0 2 2 3 】

(付記 1 9)

目標処理時間を設定するステップと、

前記目標処理時間を用いて目標並列効率を計算し、記憶装置に格納するステップと、

前記目標並列効率の妥当性を確認するステップと、

をさらに含む付記 1 乃至 1 4 のいずれか 1 つ記載の並列効率計算方法。

【 0 2 2 4 】

(付記 2 0)

前記目標並列効率の妥当性が確認された場合には、チューニング実施後の並列効率を計算し、記憶装置に格納するステップと、

前記チューニング実施後の並列効率と前記目標並列効率とを比較するステップと、

をさらに含む付記 1 9 記載の並列効率計算方法。

【 0 2 2 5 】

(付記 2 1)

目標処理時間を設定するステップと、

異なるアルゴリズム毎に当該アルゴリズムにおける並列効率を用いて必要となるプロセッサ数の見積値を計算し、記憶装置に格納するステップと、

前記プロセッサ数の見積値が前記並列計算機システムにおいて実施する当該アルゴリズムによる処理の並列化による処理時間の短縮度合いの向上の限度を表す加速率より小さく且つ異なるアルゴリズムについて計算された前記プロセッサ数の見積値のうち最小の値となるアルゴリズムを抽出するステップと、

をさらに含む付記 1 乃至 1 4 のいずれか 1 つ記載の並列効率計算方法。

【 0 2 2 6 】

(付記 2 2)

付記 1 乃至 2 1 のいずれか 1 つ記載の並列効率計算方法をコンピュータに実行させるためのプログラム。

【 0 2 2 7 】

(付記 2 3)

並列計算機システムの並列効率を計算する並列効率計算装置であって、

前記並列計算機システムに含まれる各プロセッサ間の負荷の均衡度合いを表すロードバランス寄与率を計算し、記憶装置に格納するロードバランス寄与率計算手段と、

前記並列計算機システムにおいて実施した処理のうち各プロセッサにより並列計算された部分の、時間についての割合を表す仮想並列化率を計算し、記憶装置に格納する仮想並列化率計算手段と、

前記並列計算機システムに含まれる全プロセッサの全処理時間に対する各並列性能阻害要因部分の処理時間の割合を表す並列性能阻害要因寄与率を計算し、記憶装置に格納する並列性能阻害要因寄与率計算手段と、

前記ロードバランス寄与率と前記仮想並列化率と前記並列性能阻害要因寄与率とを用いて並列効率を計算し、記憶装置に格納する手段と、
を有する並列効率計算装置。

【 0 2 2 8 】

(付記 2 4)

並列計算機システムの並列効率を計算する並列効率計算装置であって、
前記並列計算機システムに含まれる各プロセッサ間の負荷の均衡度合いを表すロードバランス寄与率を計算し、記憶装置に格納するロードバランス寄与率計算手段と、

前記並列計算機システムにおいて実施する処理の並列化による処理時間の短縮度合いの向上の限度を表す加速率を計算し、記憶装置に格納する加速率計算手段と、

前記並列計算機システムに含まれる全プロセッサの全処理時間に対する各並列性能阻害要因部分の処理時間の割合を表す並列性能阻害要因寄与率を計算し、記憶装置に格納する並列性能阻害要因寄与率計算手段と、

前記ロードバランス寄与率と前記加速率と前記並列性能阻害要因寄与率とを用いて並列効率を計算し、記憶装置に格納する手段と、

を有する並列効率計算方法。

【 0 2 2 9 】

(付記 2 5)

並列計算機システムの並列効率を計算する並列効率計算装置であって、
前記並列計算機システムに含まれる各プロセッサ間の負荷の均衡度合いを表すロードバランス寄与率を計算し、記憶装置に格納するロードバランス寄与率計算手段と、

前記並列計算機システムに含まれる全プロセッサの全処理時間に対する各並列性能阻害要因部分の処理時間の割合を表す並列性能阻害要因寄与率を計算し、記憶装置に格納する並列性能阻害要因寄与率計算手段と、

前記ロードバランス寄与率と前記並列性能阻害要因寄与率とを用いて並列効率を計算し、記憶装置に格納する手段と、

を有する並列効率計算装置。

【 0 2 3 0 】

(付記 2 6)

並列計算機システムの並列効率を計算する並列効率計算装置であって、

前記並列計算機システムに含まれる各プロセッサ間の負荷の均衡度合いを表すロードバランス寄与率を計算し、記憶装置に格納するロードバランス寄与率計算手段と、

前記並列計算機システムにおいて実施する処理のうち各プロセッサにより並列計算される部分の、時間についての割合を表す仮想並列化率を計算し、記憶装置に格納する仮想並列化率計算手段と、

前記並列計算機システムに含まれる各プロセッサにおいて実施された処理のうち並列計算部分の処理時間の和と、前記各プロセッサにおいて実施された処理の処理時間の和と、前記ロードバランス寄与率と、前記仮想並列化率とを用いて並列効率を計算し、記憶装置に格納する手段と、

を有する並列効率計算装置。

【 0 2 3 1 】

(付記 2 7)

並列計算機システムの並列効率を計算する並列効率計算装置であって、

1 プロセッサにより処理を実施する場合において当該処理のうち並列性能阻害部分の全処理時間に相当する第 1 の処理時間を計算し、記憶装置に格納する手段と、

前記並列計算機システムに含まれる各プロセッサにおいて実施された処理のうち並列計算部分の処理時間の和である第 2 の処理時間を計算し、記憶装置に格納する手段と、

前記並列計算機システムにおいて使用したプロセッサの数と、前記並列計算機システムに含まれる各プロセッサにおいて実施された処理の処理時間のうち最長の処理時間と、前記第 1 の処理時間と、前記第 2 の処理時間とを用いて並列効率を計算し、記憶装置に格納する手段と、

を有する並列効率計算装置。

【 0 2 3 2 】

【発明の効果】

以上述べたように本発明によれば、「ロードバランスが保たれている」という条件をはずし、グリッドコンピューティングを含む、ヘテロなプロセッサ環境を含め多種類の並列処理に適用でき、並列効率と並列性能評価指標及び並列性能阻害要因間の定量的関係付けを行うことができる。

【 0 2 3 3 】

また、並列効率等を用いて、並列計算機システムの適切な運用も可能となる。

【 0 2 3 4 】

さらに、並列効率等を用いて、並列計算機システムの能力増強、更新等に対する適切な判断が可能になる。

【 0 2 3 5 】

さらに、並列効率等を用いて、並列計算機システムにおいて実行するプログラムのチューニングやアルゴリズムの選定を適切に実施できるようになる。

【図面の簡単な説明】

【図 1】

プロセッサ間でロードバランスが保たれている状態を表す図である。

【図 2】

各プロセッサにおける処理時間の分類例を示す図である。

【図 3】

プロセッサ間でロードバランスが保たれていない状態（4つのプロセッサを割り当てて、その中のプロセッサの1つで処理する場合）を表す図である。

【図 4】

r_1 と $r_i(p)$ の関係のモデル化の一例を示す図である。

【図 5】

CPU性能にばらつきがあり且つデータパラレル処理を行っている状態を表す図である。

【図 6】

（a）は1プロセッサで処理する場合の処理時間を表す図であり、（b）は4

プロセッサで処理する場合の処理時間を表す図である。

【図 7】

(a) は並列処理部 γ と通信部 χ_C の時間を考慮した場合における処理時間を表す図であり、(b) はさらに立ち上がり時間 χ_{TC} を考慮に入れた場合における処理時間を表す図である。

【図 8】

並列性能阻害要因を追加した場合における並列性能評価指標の変化を表すための図である。

【図 9】

プロセッサ間でロードバランスは保たれているが、各処理時間のロードバランスまでは保たれていない場合の例を表す図である。

【図 10】

1 つのプロセッサで処理する場合の処理時間を表す図である。

【図 11】

プロセッサ間でロードバランスが保たれていない場合の例を示す図である。

【図 12】

高並列化された場合に顕在化する並列性能阻害要因の存在を表すための図である。

【図 13】

並列性能評価指標の計算例を表す図である。

【図 14】

稼働時間と処理時間の総和の関係を表す図である。

【図 15】

稼働時間と処理時間と並列効率を考慮した処理時間との関係を表す図である。

【図 16】

データ並列を分散メモリ並列計算機システムで実施した場合の処理時間の例を表す図である。

【図 17】

原状の CPU 性能に基づく並列性能評価指標と CPU 性能が 5 倍のシステムに

入れ替えた場合の推定並列性能評価指標とを比較するための図である。

【図 1 8】

CPU性能が5倍のシステムに入れ替えた場合における試算のためのデータを表す図である。

【図 1 9】

本発明の一実施の形態に係る機能ブロック図である。

【図 2 0】

サンプリングによる事象発生の確認及びカウントを表す概念図である。

【図 2 1】

表1のプログラム実行時のサンプリング結果例を表す図である。

【図 2 2】

並列性能分析装置の処理フローの一例を表す図である。

【図 2 3】

時間測定による処理時間の測定結果例を表す図である。

【図 2 4】

サンプリングによる処理時間の測定結果例を表す図である。

【図 2 5】

プロセッサ数最適化処理の処理フローの第1の部分の一例を表す図である。

【図 2 6】

プロセッサ数最適化処理の処理フローの第2の部分の一例を表す図である。

【図 2 7】

プロセッサ増設見積処理の処理フローの一例を表す図である。

【図 2 8】

システムリプレイスデータ処理の処理フローの一例を表す図である。

【図 2 9】

CPU性能が5倍で目標並列効率が0.6の場合の通信についての性能指針を表すための図である。

【図 3 0】

システム運用効率向上処理のための処理フローの一例を示す図である。

【図 3 1】

チューニング処理の処理フローの一例を示す図である。

【図 3 2】

チューニング前と 1 回目のチューニングを実施した後の並列性能評価指標の変化を表す図である。

【図 3 3】

並列処理に向かないアルゴリズムに基づく並列処理プログラムによる処理時間を表す図である。

【図 3 4】

並列処理に向くアルゴリズムに基づく並列処理プログラムによる処理時間を表す図である。

【図 3 5】

並列処理に向かないアルゴリズムと並列処理に向くアルゴリズムの並列性能指標の比較等のための図である。

【図 3 6】

アルゴリズム選定処理の処理フローを表す図である。

【図 3 7】

ある並列処理システムのログデータの一例を示す図である。

【図 3 8】

実施例 1 における処理時間の測定結果を表す図である。

【図 3 9】

実施例 1 における並列性能評価指標の計算結果を表す図である。

【図 4 0】

実施例 2 における処理時間の測定結果を表す図である。

【図 4 1】

実施例 2 における並列性能評価指標の計算結果を表す図である。

【図 4 2】

実施例 3 における処理時間の測定結果を表す図である。

【図 4 3】

実施例 3 における並列性能評価指標の計算結果を表す図である。

【図 4 4】

実施例 4 における処理時間の測定結果を表す図である。

【図 4 5】

実施例 4 における並列性能評価指標の計算結果を表す図である。

【図 4 6】

実施例 5 における処理時間の測定結果を表す図である。

【図 4 7】

実施例 5 における並列性能評価指標の計算結果を表す図である。

【図 4 8】

実施例 6（冗長処理を用いたデータパラレル）における処理時間の測定結果を表す図である。

【図 4 9】

実施例 6（冗長処理を用いたデータパラレル）における並列性能評価指標の計算結果を表す図である。

【図 5 0】

実施例 6（並列処理できない部分を特定のプロセッサで処理するデータパラレル）における処理時間の測定結果を表す図である。

【図 5 1】

実施例 6（並列処理できない部分を特定のプロセッサで処理するデータパラレル）における並列性能評価指標の計算結果を表す図である。

【図 5 2】

実施例 7 における処理時間の測定結果を表す図である。

【図 5 3】

実施例 7 における並列性能評価指標の計算結果を表す図である。

【図 5 4】

実施例 8 における処理時間の測定結果を表す図である。

【図 5 5】

実施例 8 における並列性能評価指標の計算結果を表す図である。

【図 5 6】

実施例 9 における処理時間の測定結果を表す図である。

【図 5 7】

実施例 9 における並列性能評価指標の計算結果を表す図である。

【図 5 8】

実施例 1 0 における処理時間の測定結果を表す図である。

【図 5 9】

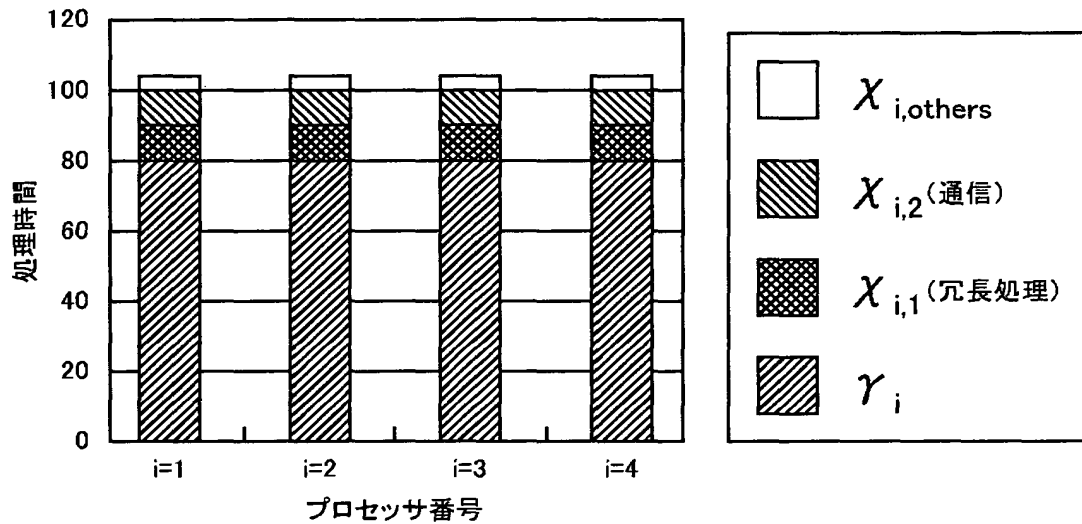
実施例 1 0 における並列性能評価指標の計算結果を表す図である。

【符号の説明】

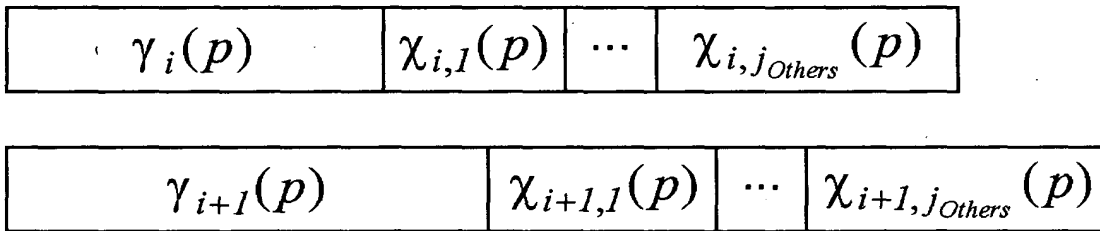
- 1 0 データ取得部 1 1 ロードバランス寄与率計算部
- 1 2 仮想並列化率計算部 1 3 並列性能阻害要因寄与率計算部
- 1 4 並列効率計算部 1 5 補助指標計算部
- 2 1 プロセッサ数最適化処理部 2 2 プロセッサ増設見積処理部
- 2 3 システムリプレイスデータ処理部
- 2 4 運用効率データ処理部 2 5 チューニング処理部
- 2 6 アルゴリズム選定処理部 2 7 並列性能評価処理部
- 3 0 ログデータ格納部
- 1 0 0 並列性能分析装置 1 1 0 出力装置
- 2 0 0 並列計算機システム 2 0 1 測定部

【書類名】 図面

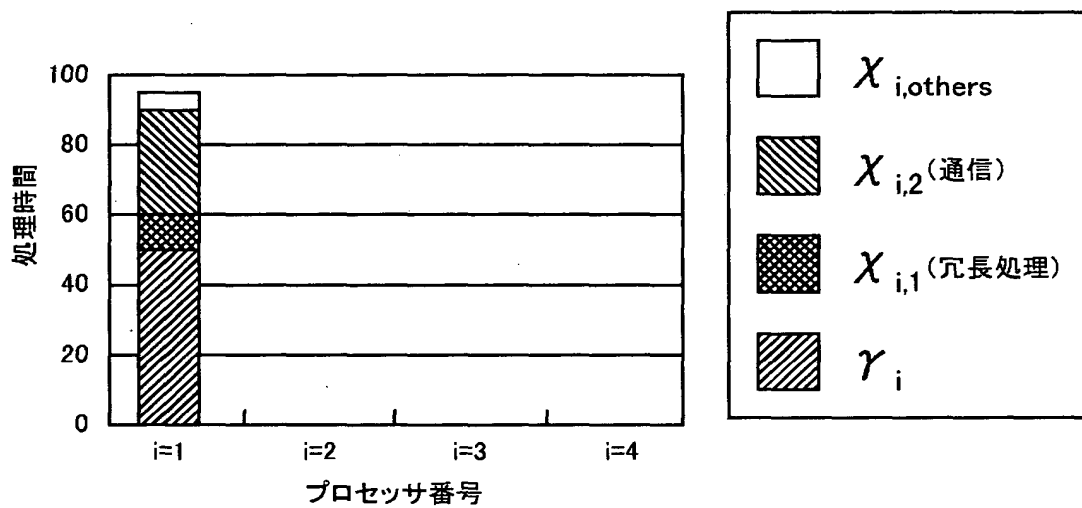
【図 1】



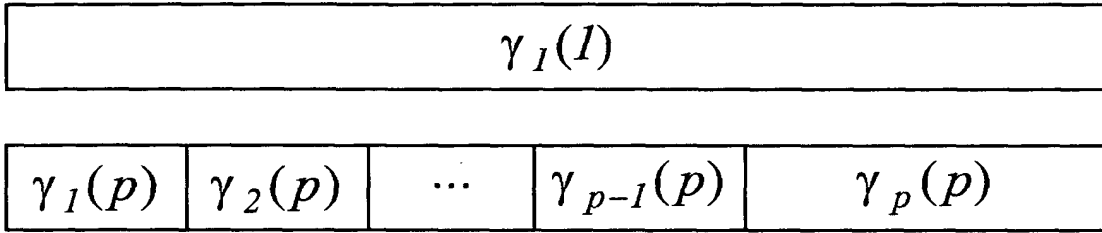
【図 2】



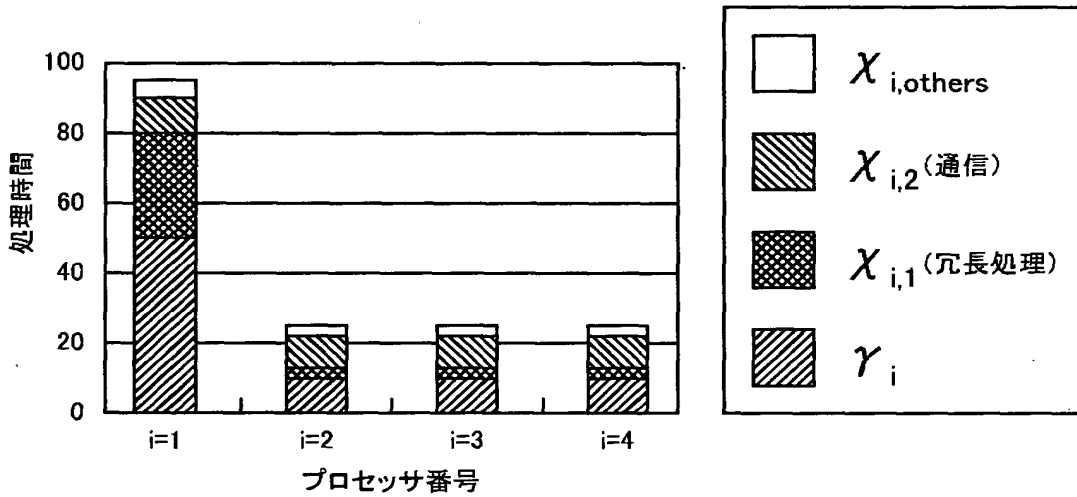
【図 3】



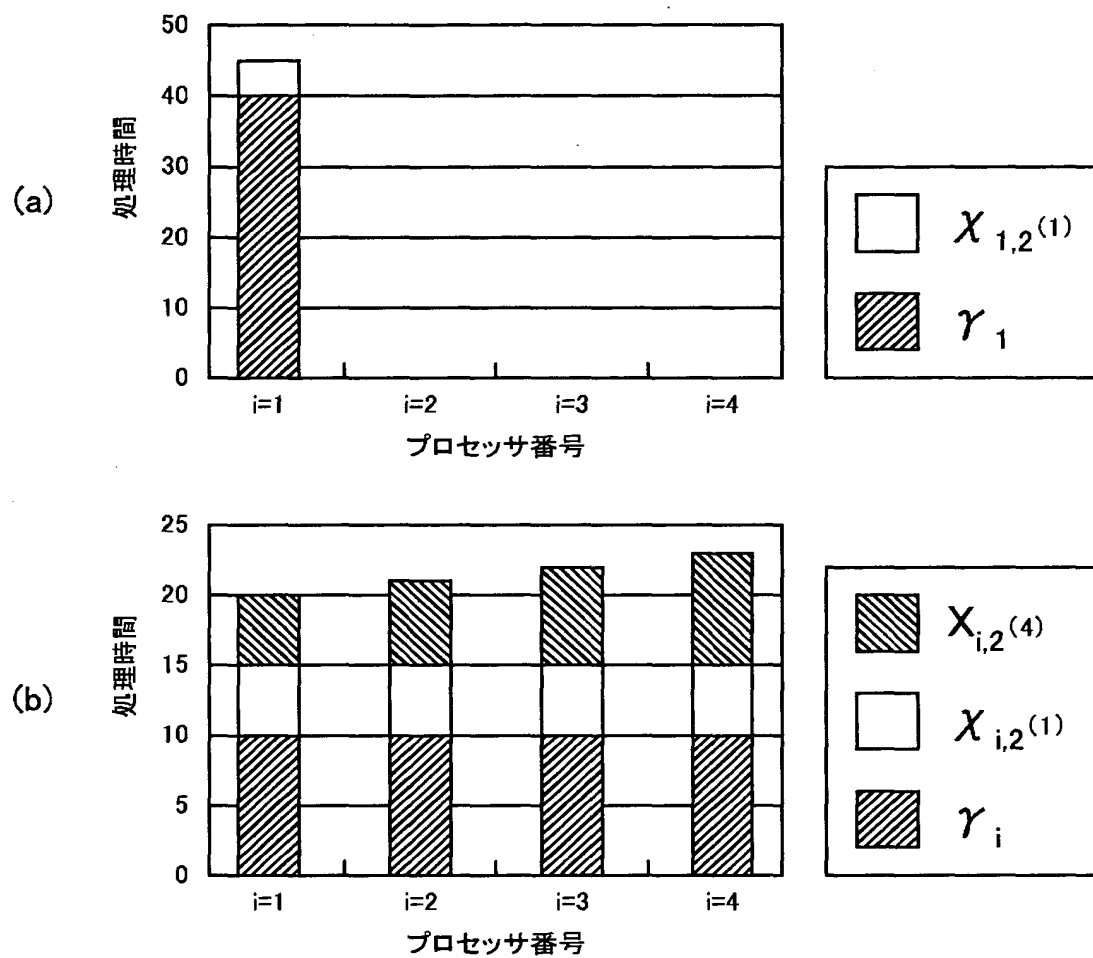
【図 4】



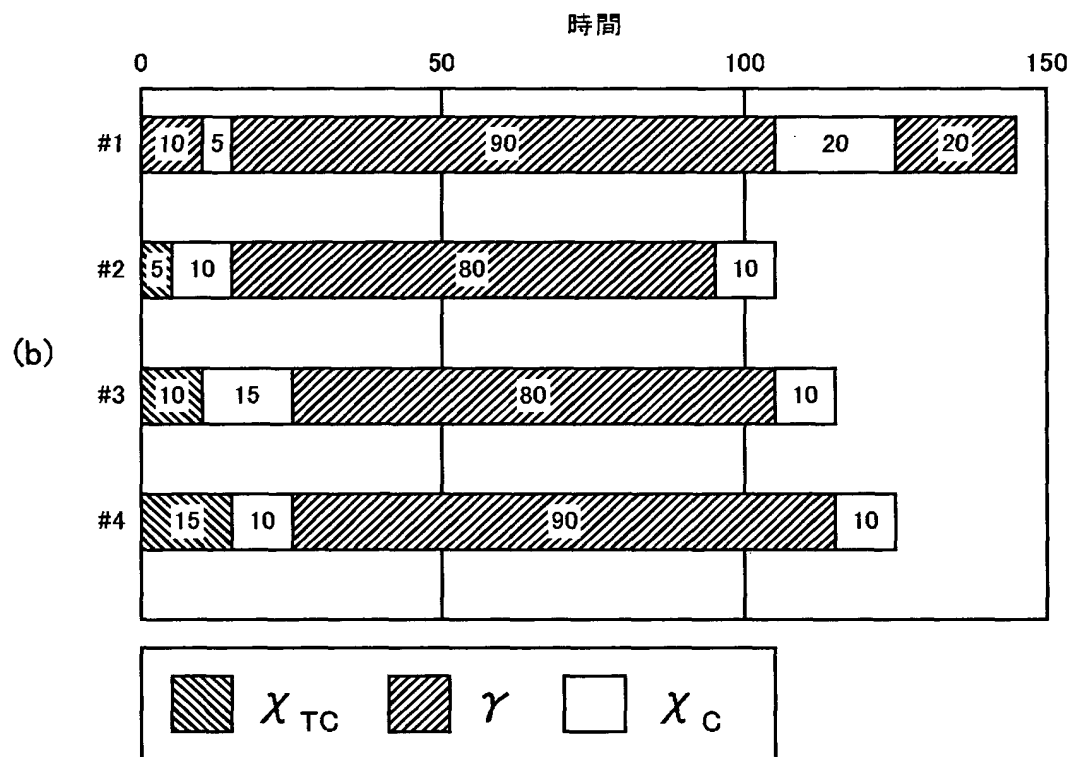
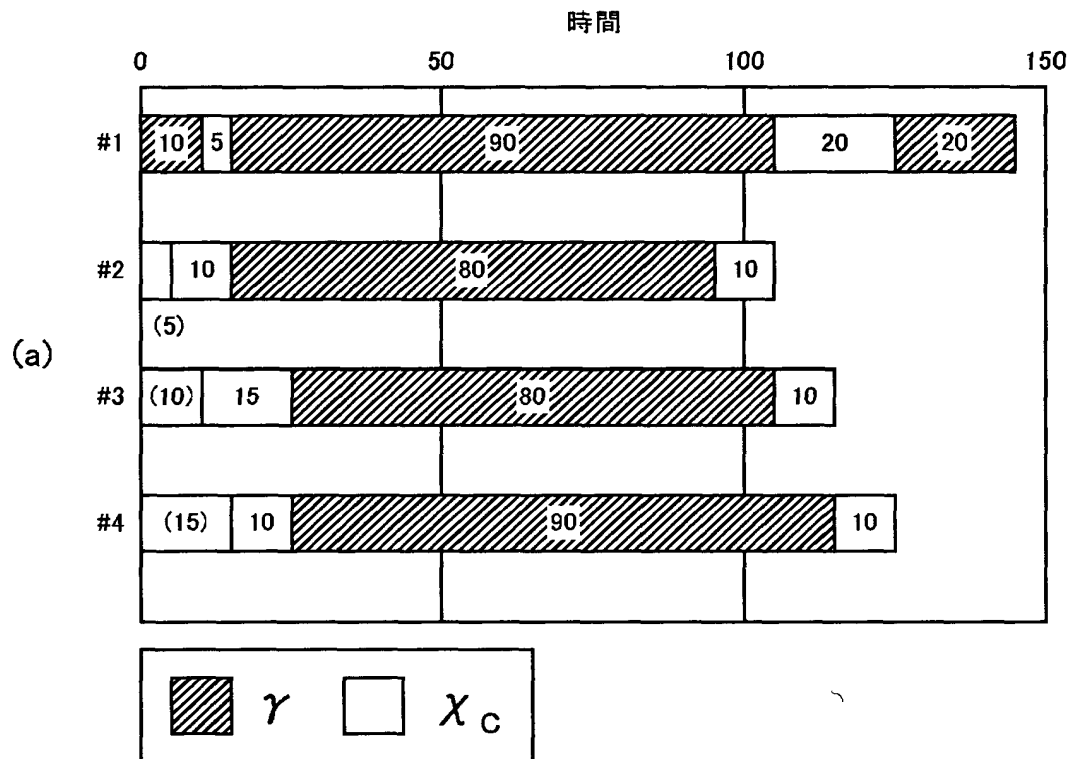
【図 5】



【図 6】



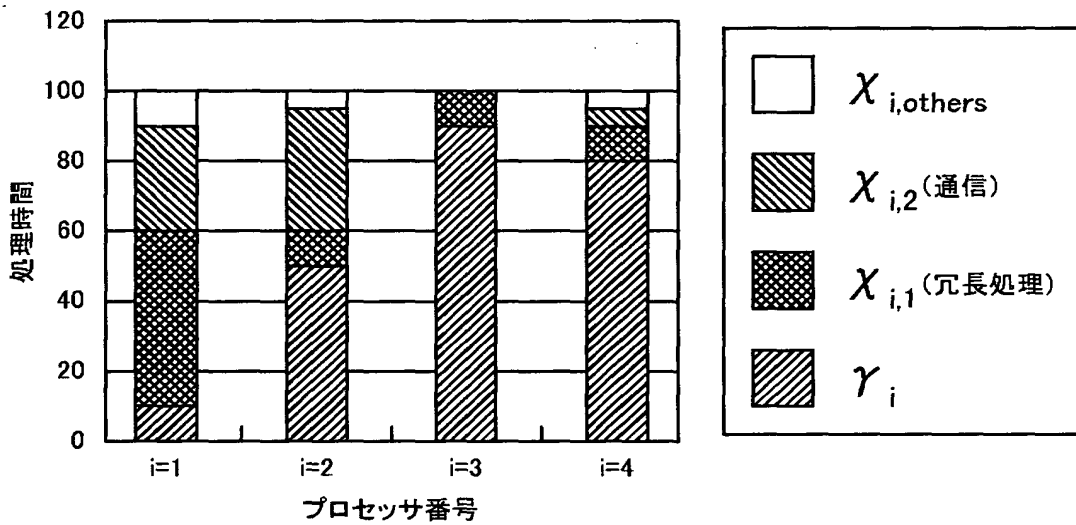
【図 7】



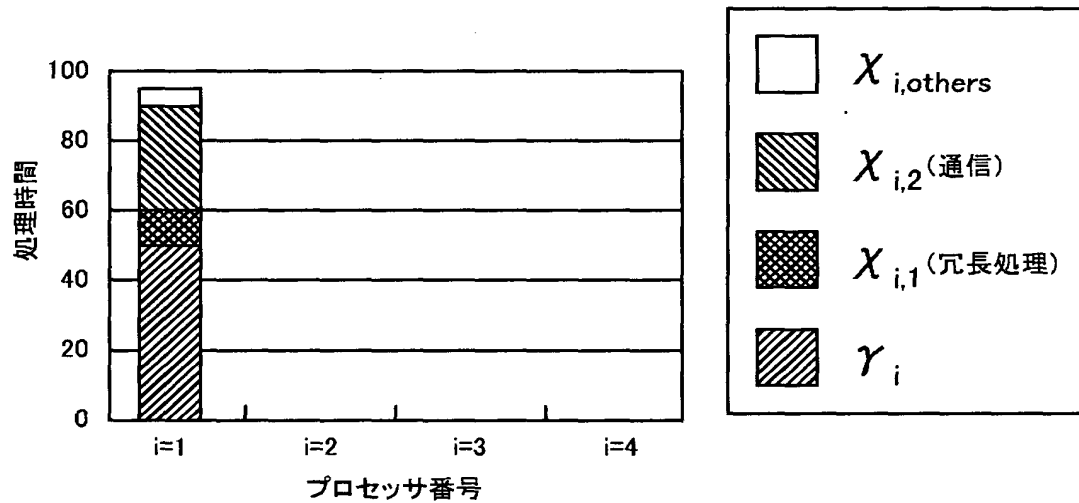
【図 8】

	$R_b(4)$	$R_C(4)$	R_{TC}	$R_p(4)$	$E_p(4)$	$E_p(4) \cdot p$
ケース1	0.7931	0.1957	-	1	0.6379	2.552
ケース2	0.8448	0.1837	0.0612	1	0.6379	2.552

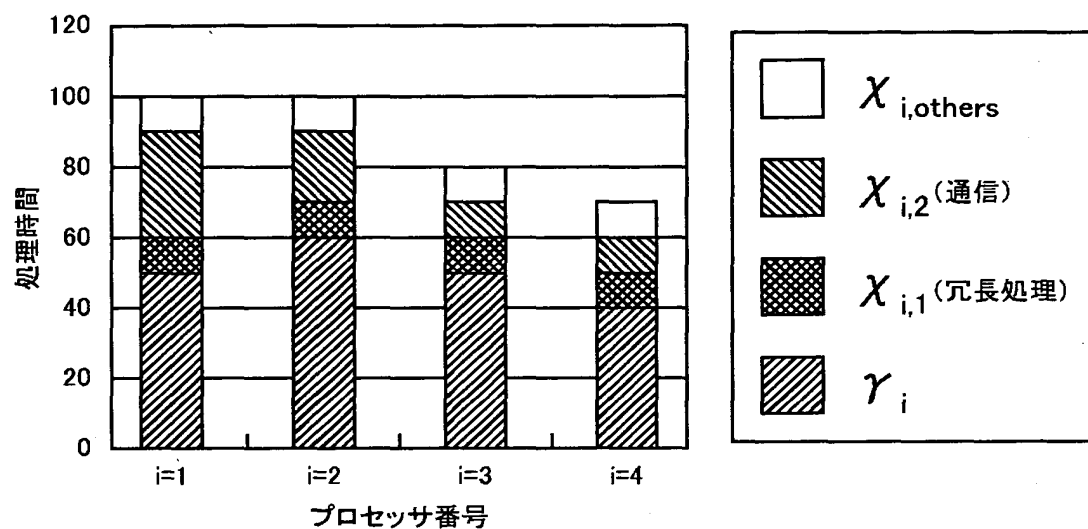
【図 9】



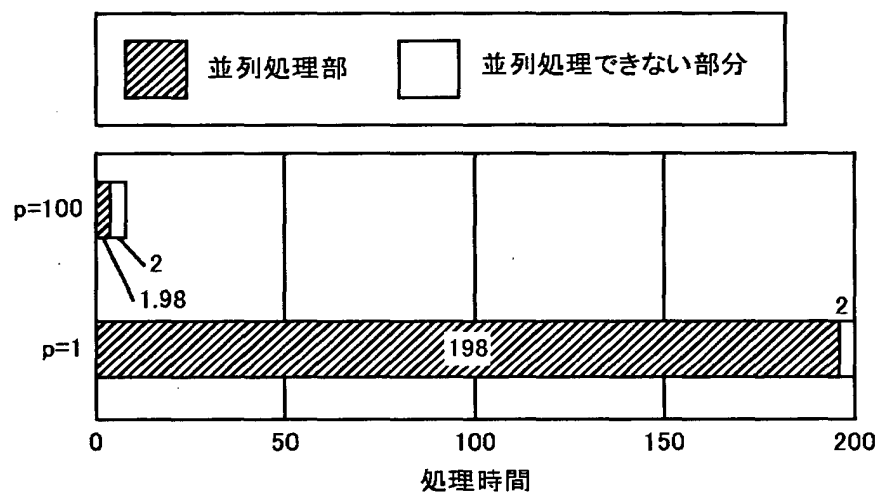
【図 10】



【図 1 1】



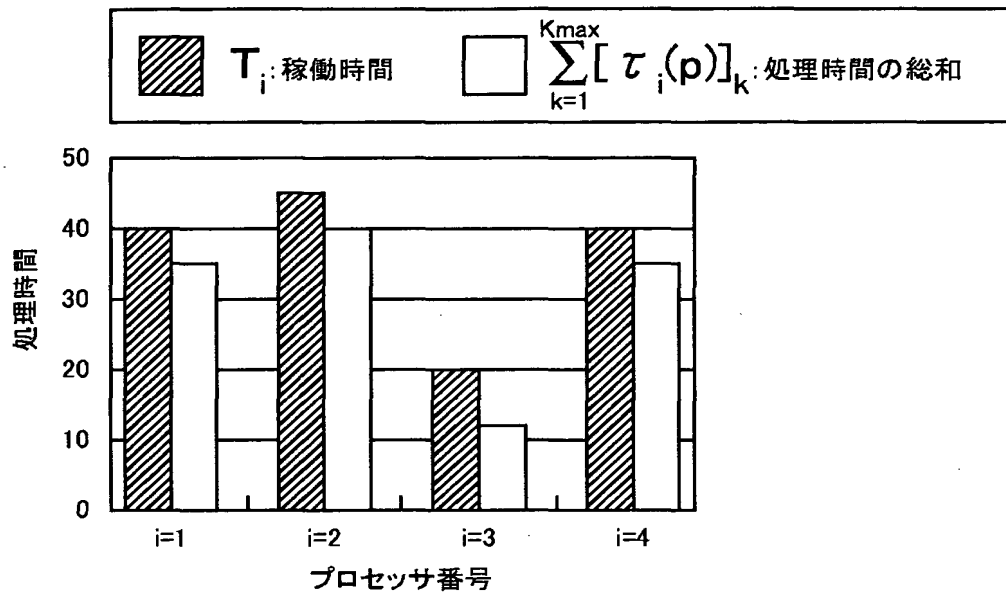
【図 1 2】



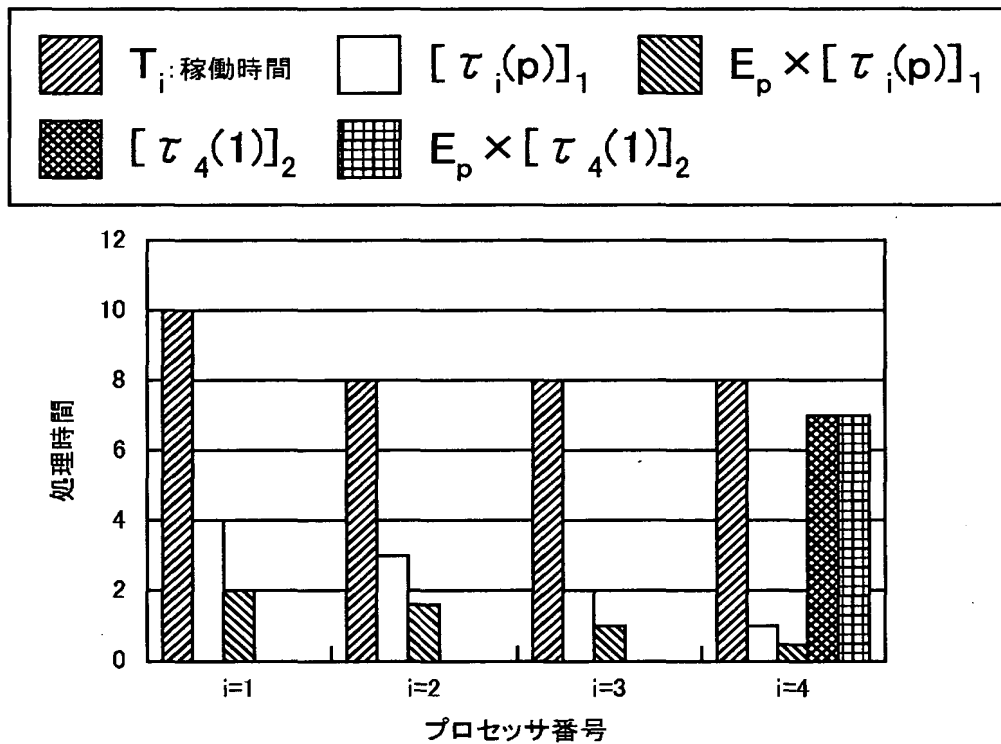
【図 1 3】

$R_b(4)$	$R_p(4)$	$A_p(p)$	$R_{RED}(4)$	$R_C(4)$	$R_{Others}(4)$	$E_p(4)$	$E_p(4) \cdot p$
0.9392	0.8821	8.482	0.2230	0.3309	0.0288	0.4443	1.777

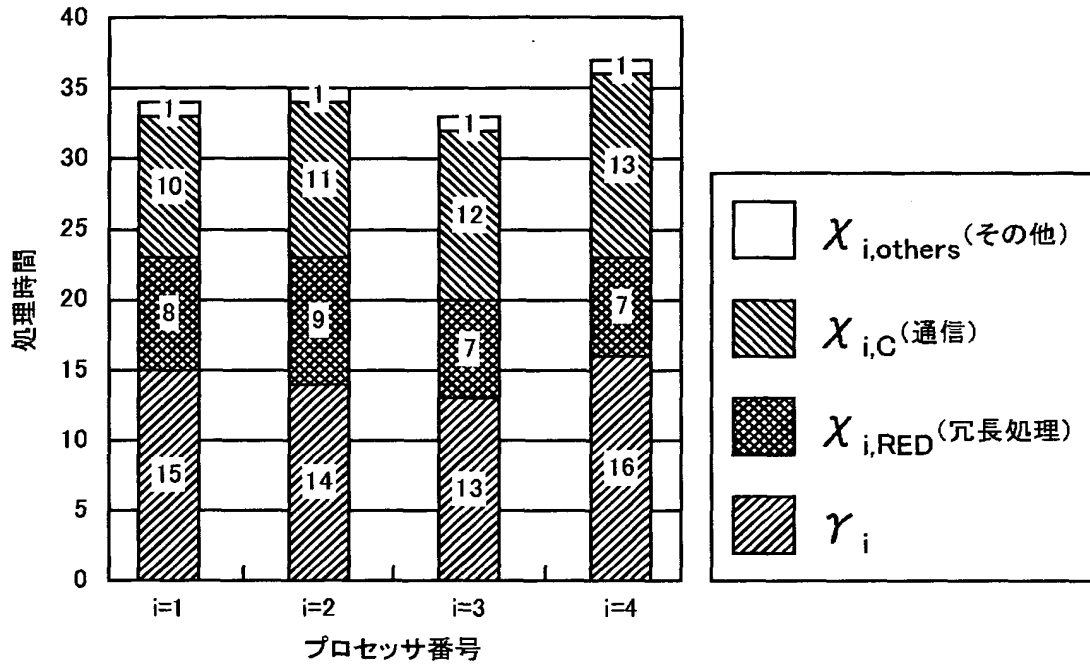
【図 1 4】



【図 1 5】



【図 1 6】



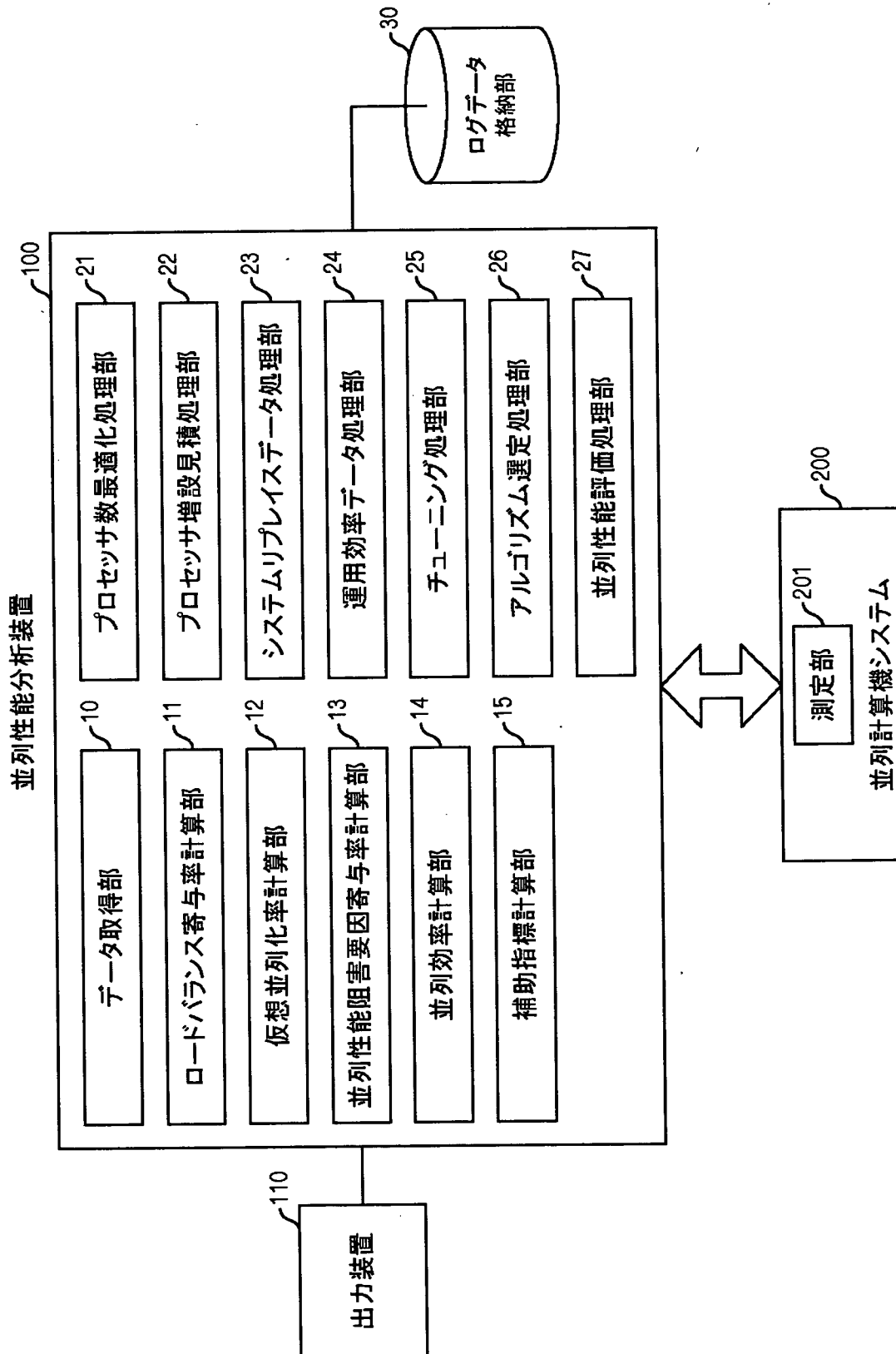
【図 1 7】

CPU性能	$R_b(4)$	$R_p(4)$	$A_p(p)$	$R_{RED}(4)$	$R_C(4)$	$R_{Others}(4)$	$E_p(4)$	$E_p(4) \cdot p$
1倍 (実測値)	0.9392	0.8821	8.482	0.2230	0.3309	0.0288	0.4443	1.777
5倍 (推定値)	0.9073	0.8821	8.482	0.0960	0.7121	0.0124	0.1846	0.7384

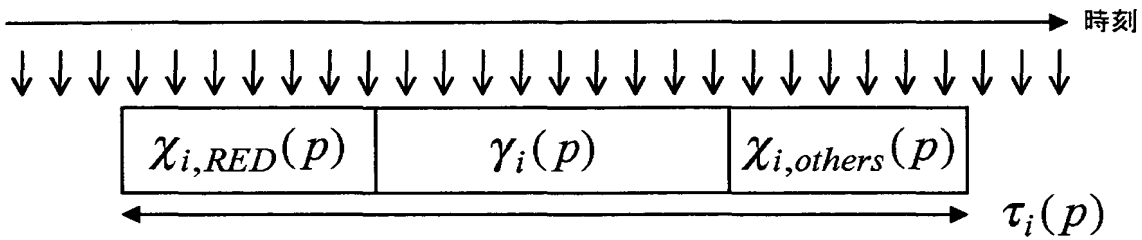
【図 1 8】

処理No	$E_p(4)$	$\sum_{i=1}^p \tau_i(p)$	p
1	0.1846	64.6	4
2	0.7219	2000.3	10
3	0.3000	512.1	2
4	1	1000	1

【図 1 9】



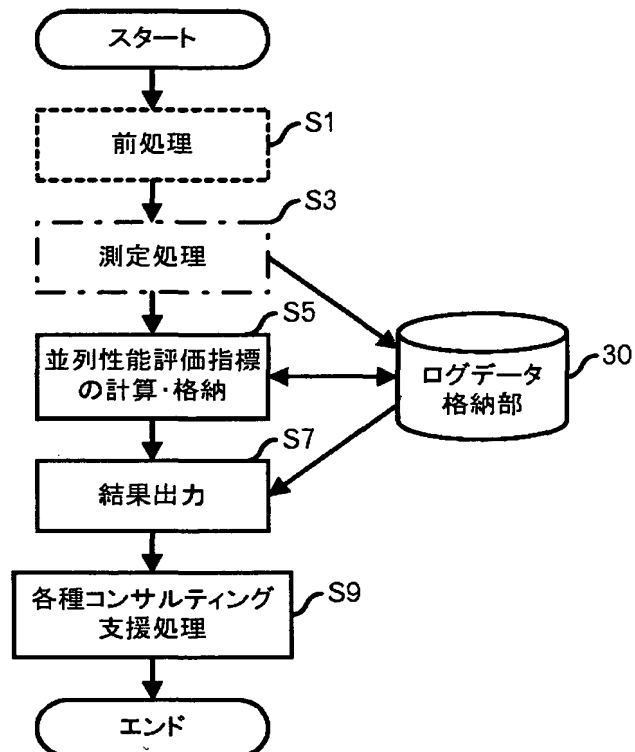
【図 2 0】



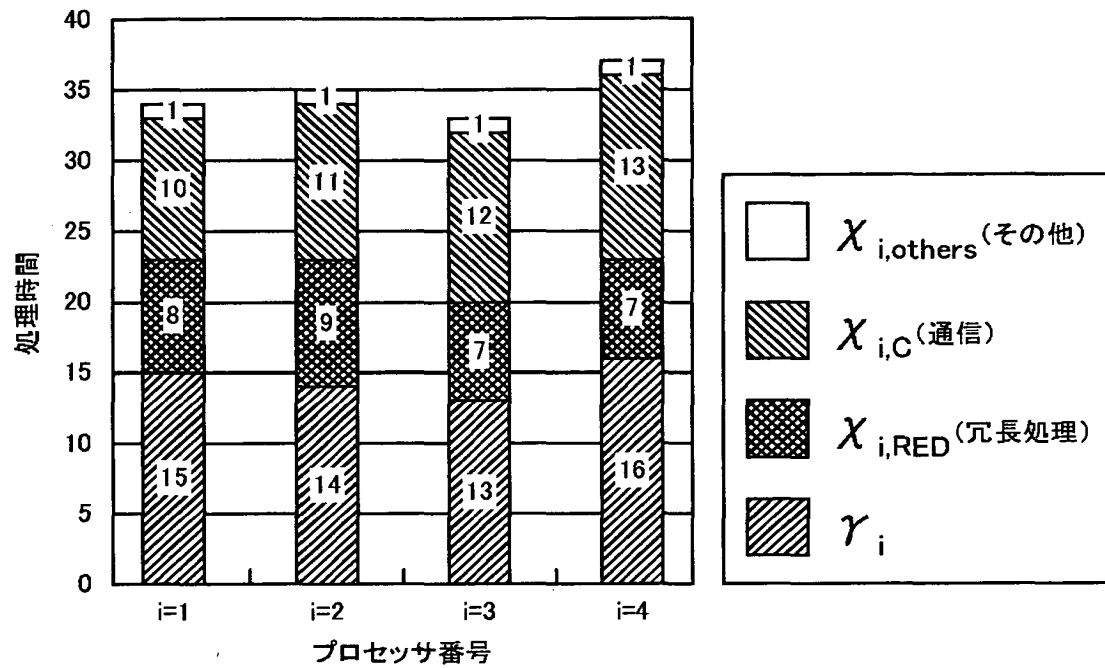
【図 2 1】

No.	(RED),1	(RED),2	(C),1	(C),2	(RED),1+(RED),2	(C),1+(C),2
#1	10	40	100	99	50	199
#2	11	40	101	100	51	201
#3	10	38	104	100	48	204
#4	9	39	98	98	48	196

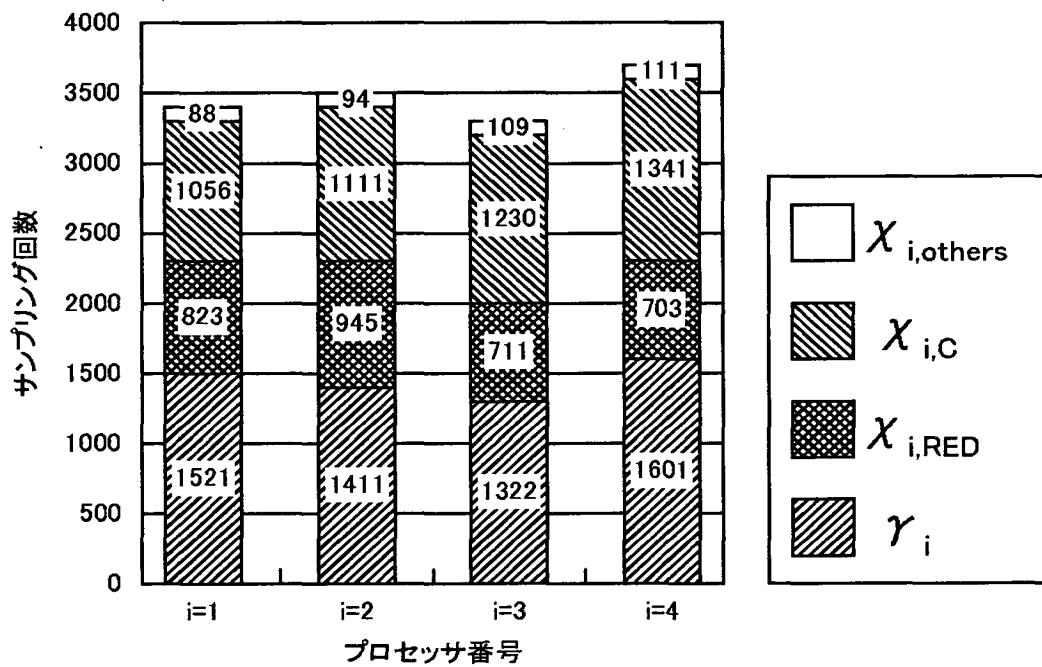
【図 2 2】



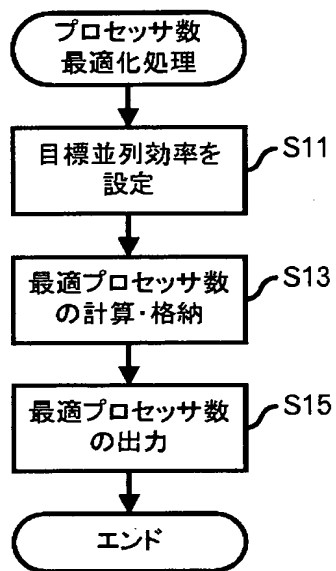
【図 2 3】



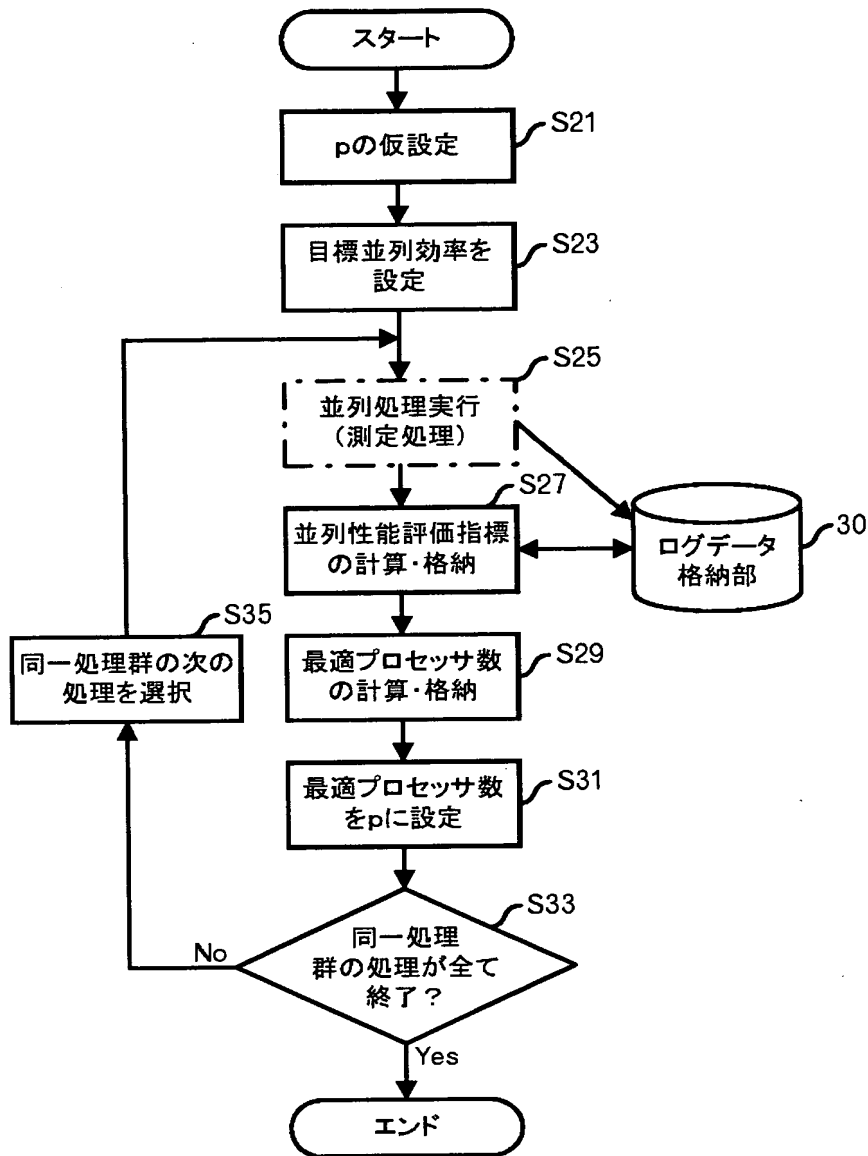
【図 2 4】



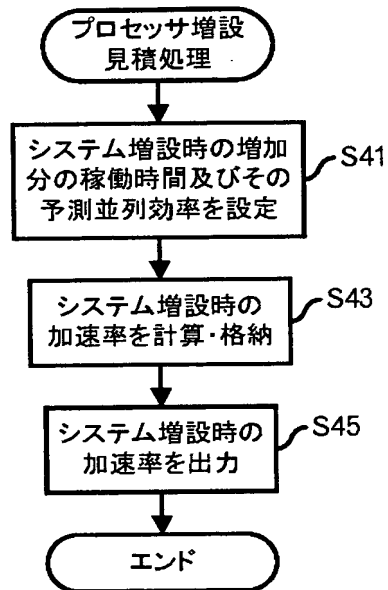
【図 2 5】



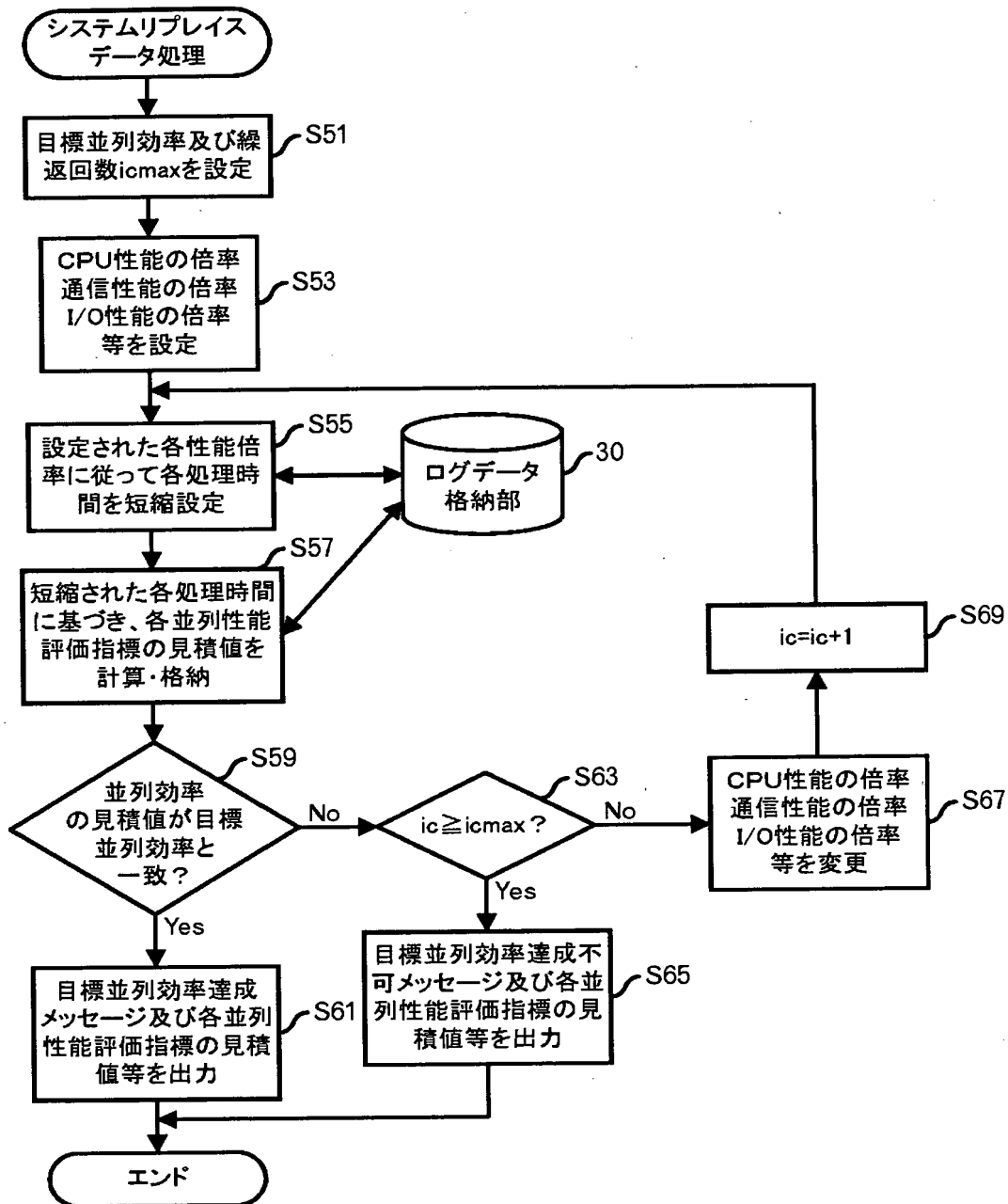
【図 2 6】



【図 2 7】



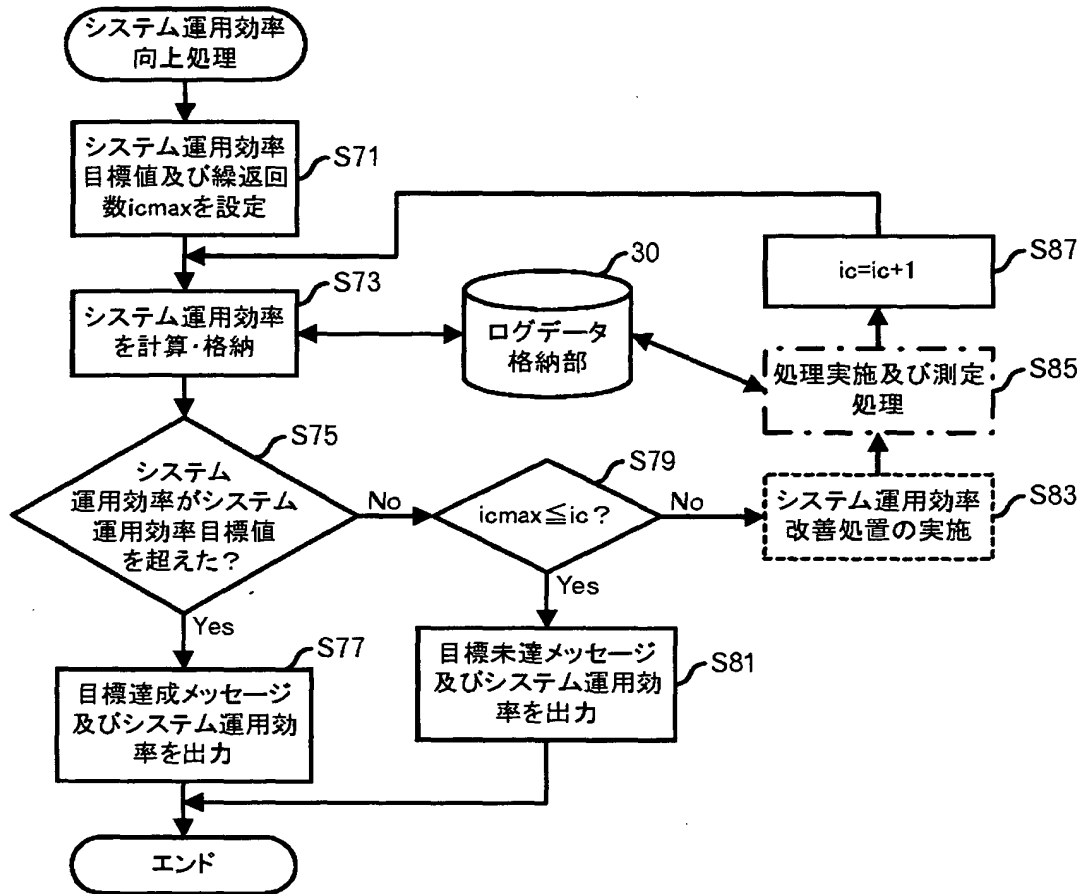
【図 28】



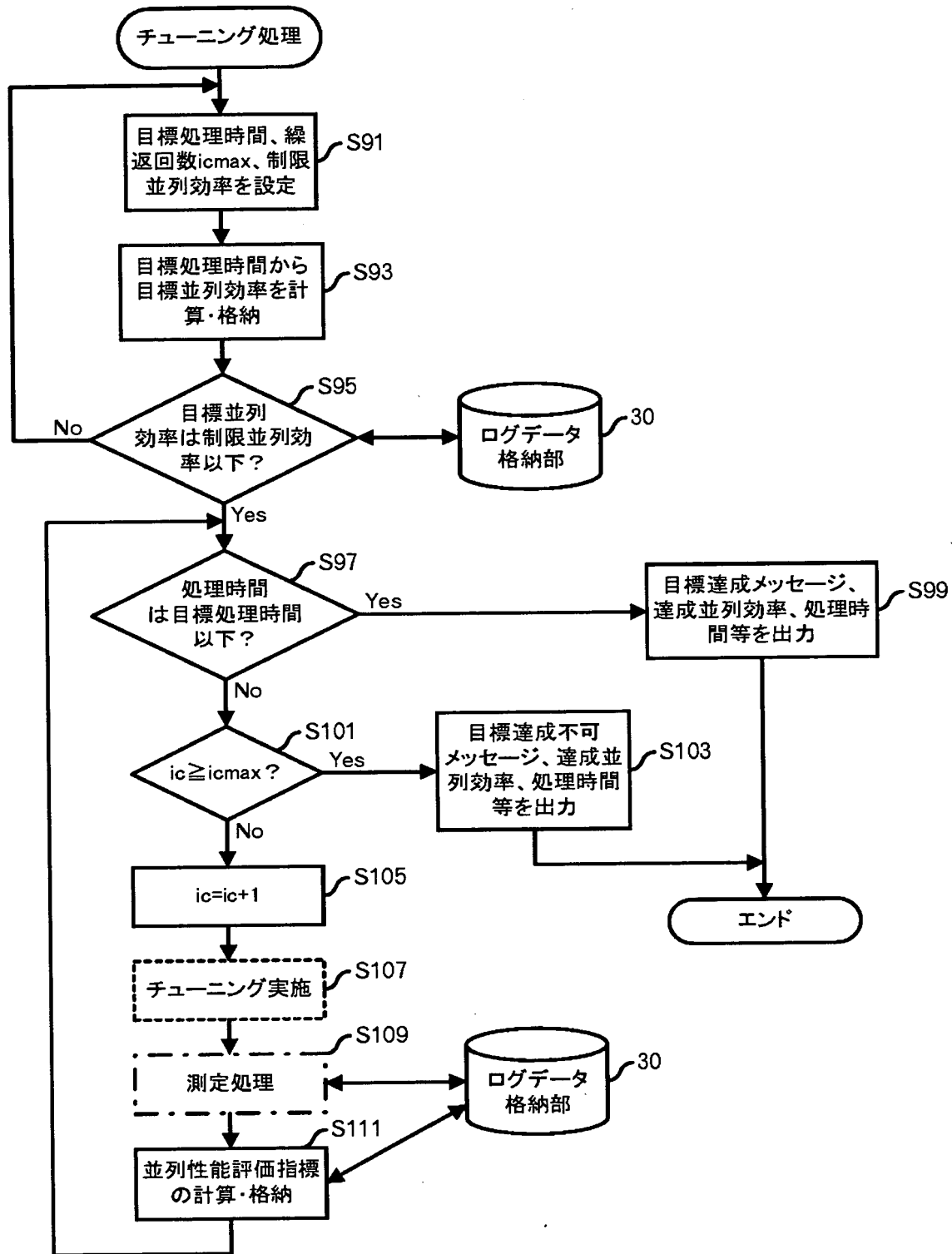
【図 29】

倍率	$(R_b(4))_E$	$(R_p(4))_E$	$(A_p(4))_E$	$(R_{RED}(4))_E$	$(R_C(4))_E$	$(R_{others}(4))_E$	$(E_p(4))_E$	$(E_p(4))_E \cdot p$
$A_{CPU}=5$ $A_C=\infty$	0.9688	0.8821	8.482	0.3333	0	0.0430	0.6850	2.740
$A_{CPU}=5$ $A_C=19.2$	0.9583	0.8821	8.482	0.2953	0.1141	0.0381	0.6002	2.401

【図30】



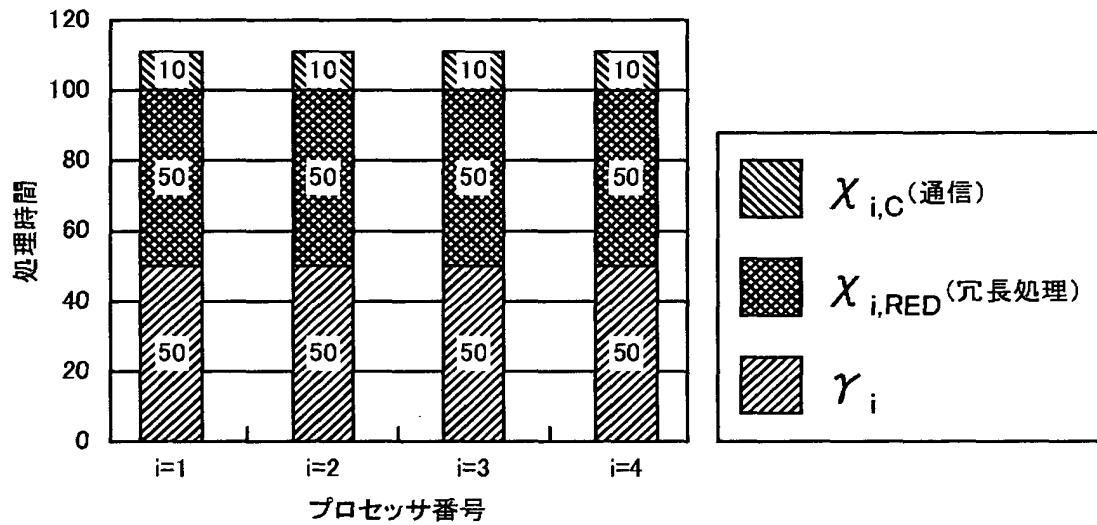
【図 3 1】



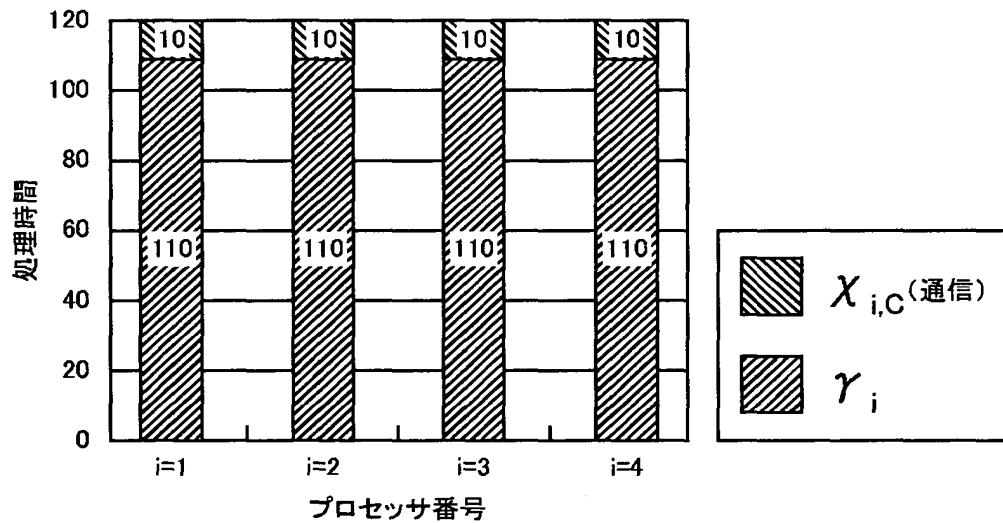
【図 3 2】

チューニング	$R_b(4)$	$R_p(4)$	$A_p(4)$	$R_{RED}(4)$	$R_C(4)$	$R_{Others}(4)$	$E_p(4)$	$\max(\tau_i)$
前	0.9392	0.8821	8.482	0.2230	0.3309	0.0288	0.4443	37
一回目後	0.9508	0.8821	8.482	0.2672	0.1983	0.0345	0.5389	30.5

【図 3 3】



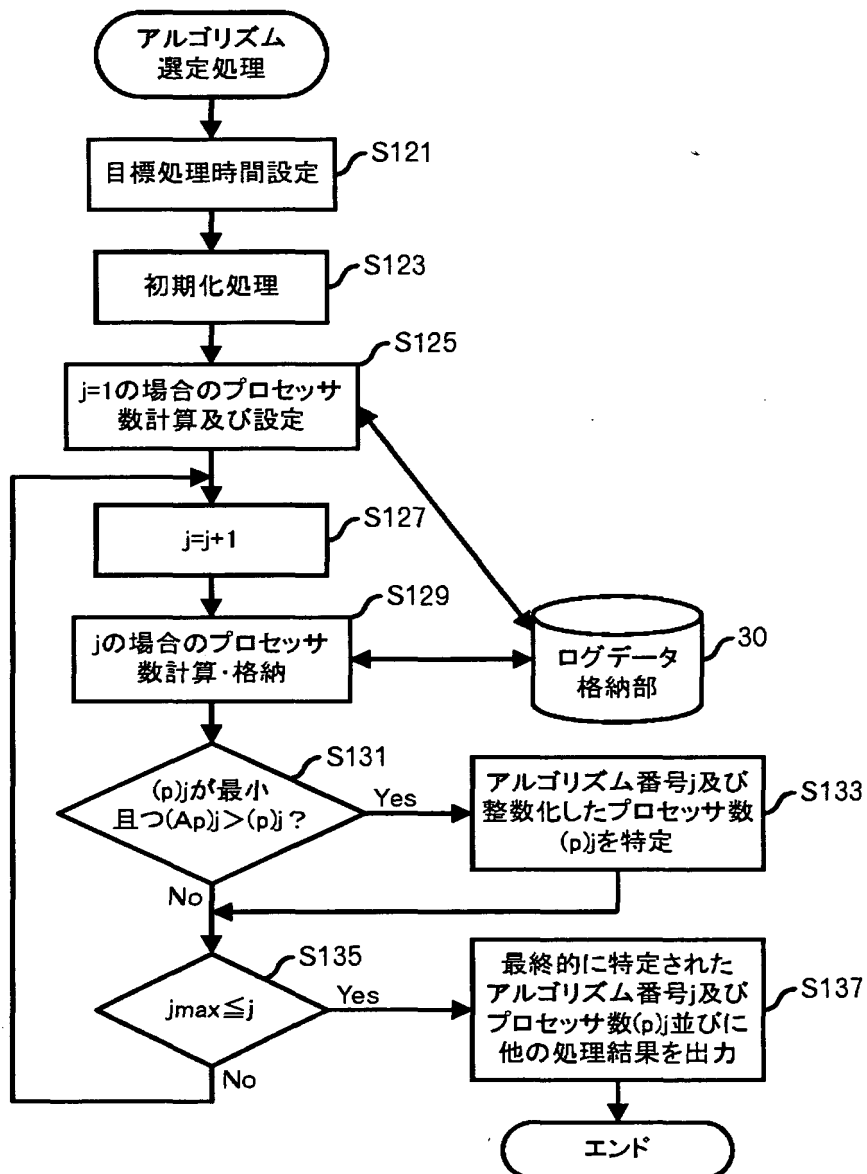
【図 3 4】



【図 3 5】

j	アルゴリズム	$A_p(4)$	$E_p(4)$	τ	$(p)_j$	上限値以下か $A_p(4) > (p)_j$	$(p)_T$
1	並列処理に向かない	5.000	0.5682	110	7.782	×	—
2	並列処理に向く	∞	0.9167	120	6.618	○	7

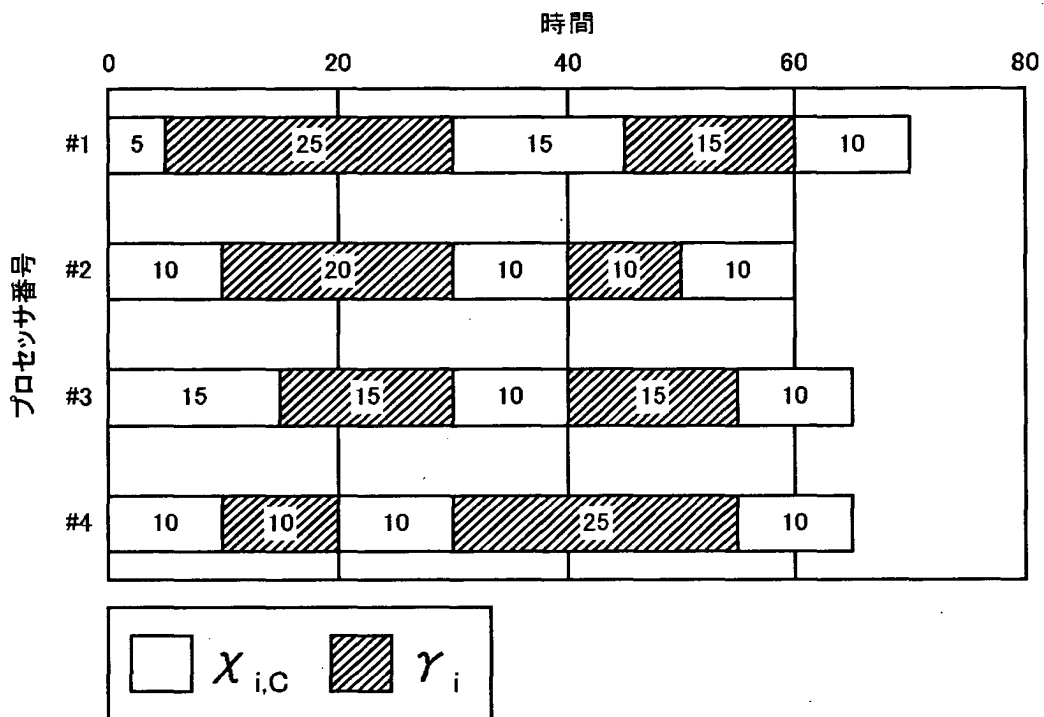
【図 3 6】



【図 3 7】

処理No.	$R_b(4)$	$R_p(4)$	$A_p(4)$	$R_{RED}(4)$	$R_C(4)$	$E_p(4)$	$E_p(4) \cdot p$
1	1.000	1.000	∞	0.0000	0.08333	0.9167	3.667
2	1.000	1.000	∞	0.0000	0.08333	0.9167	3.667
3	1.000	1.000	∞	0.0000	0.08333	0.9167	3.667
4	1.000	1.000	∞	0.0000	0.08333	0.9167	3.667
5	1.000	0.8000	5.000	0.4545	0.09091	0.5682	2.273

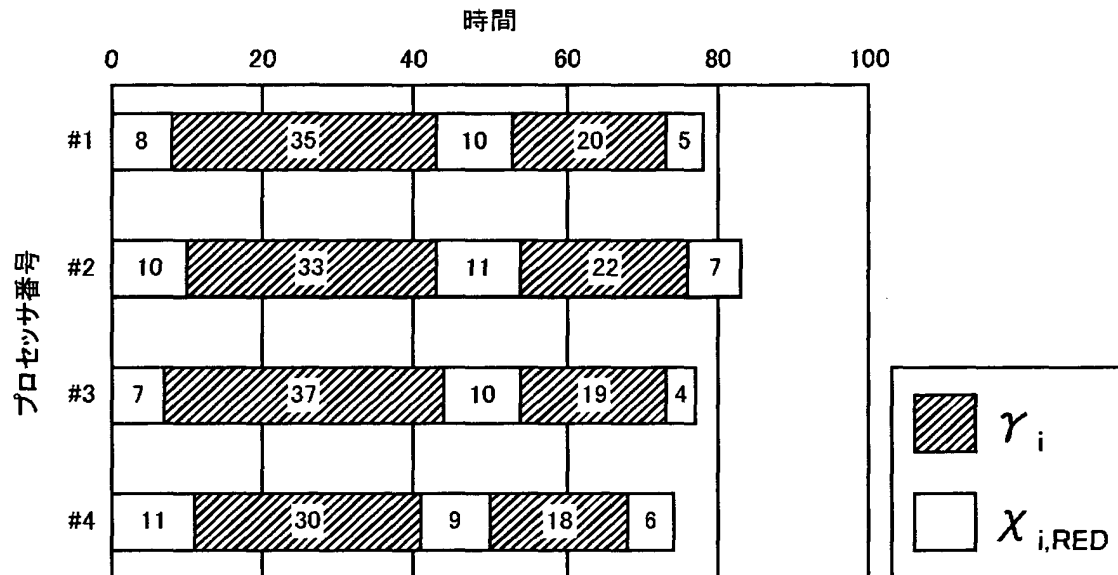
【図 3 8】



【図 3 9】

$R_b(4)$	$R_p(4)$	$A_p(4)$	$R_C(4)$	$E_p(4)$	$E_p(4) \cdot p$
0.9286	1.000	∞	0.4808	0.4821	1.928

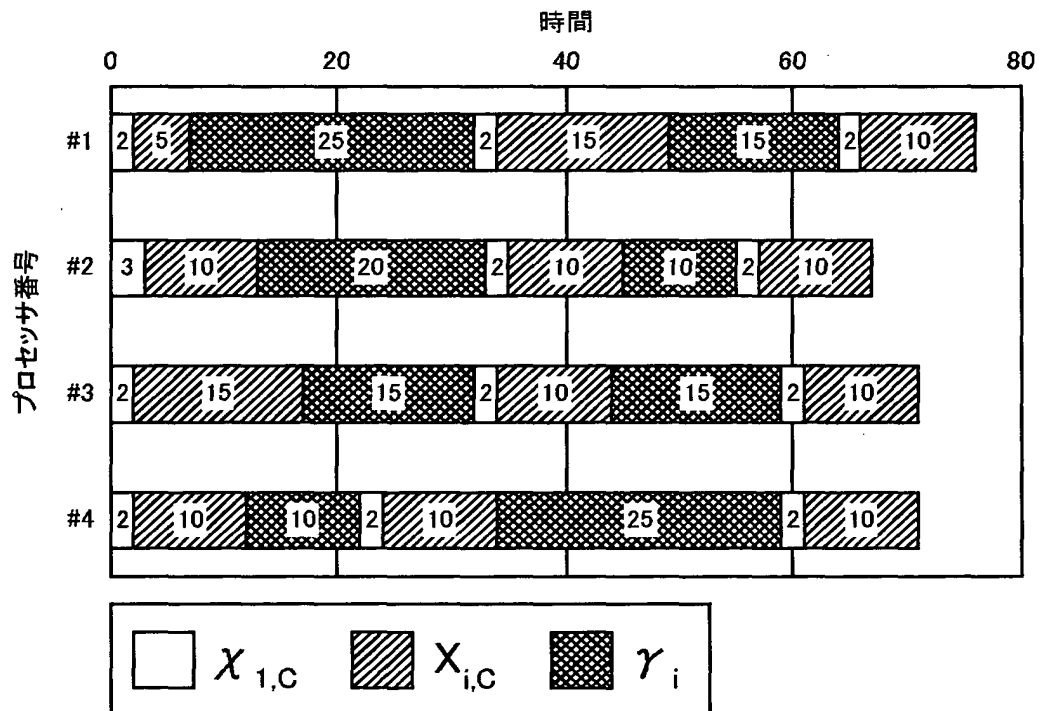
【図 4 0】



【図 4 1】

$R_b(4)$	$R_p(4)$	$A_p(4)$	$R_{RED}(4)$	$E_p(4)$	$E_p(4) \cdot p$
0.9398	0.8973	9.737	0.3141	0.7184	2.874

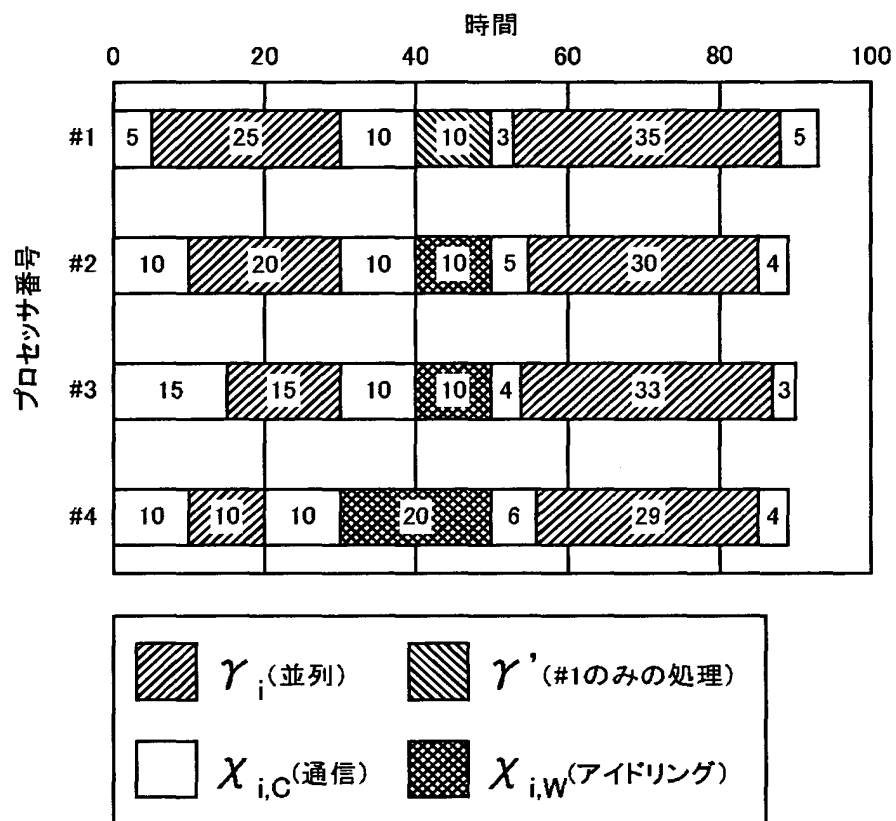
【図 4 2】



【図 4 3】

$R_b(4)$	$R_p(4)$	$A_p(4)$	$R_C(4)$	$E_p(4)$	$E_p(4) \cdot p$
0.9375	0.9557	22.57	0.5263	0.4647	1.859

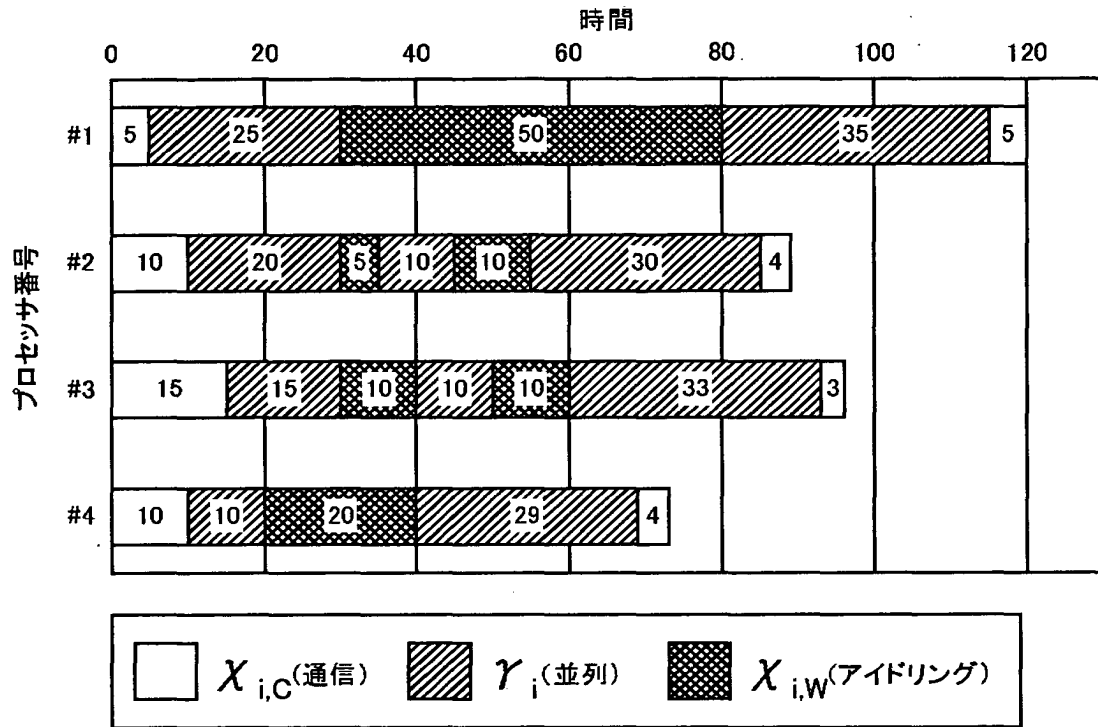
【図 4 4】



【図 4 5】

$R_b(4)$	$R_p(4)$	$A_p(4)$	$R_C(4)$	$R_W(4)$	$E_p(4)$	$E_p(4) \cdot p$
0.9704	1.000	∞	0.3158	0.1108	0.5564	2.226

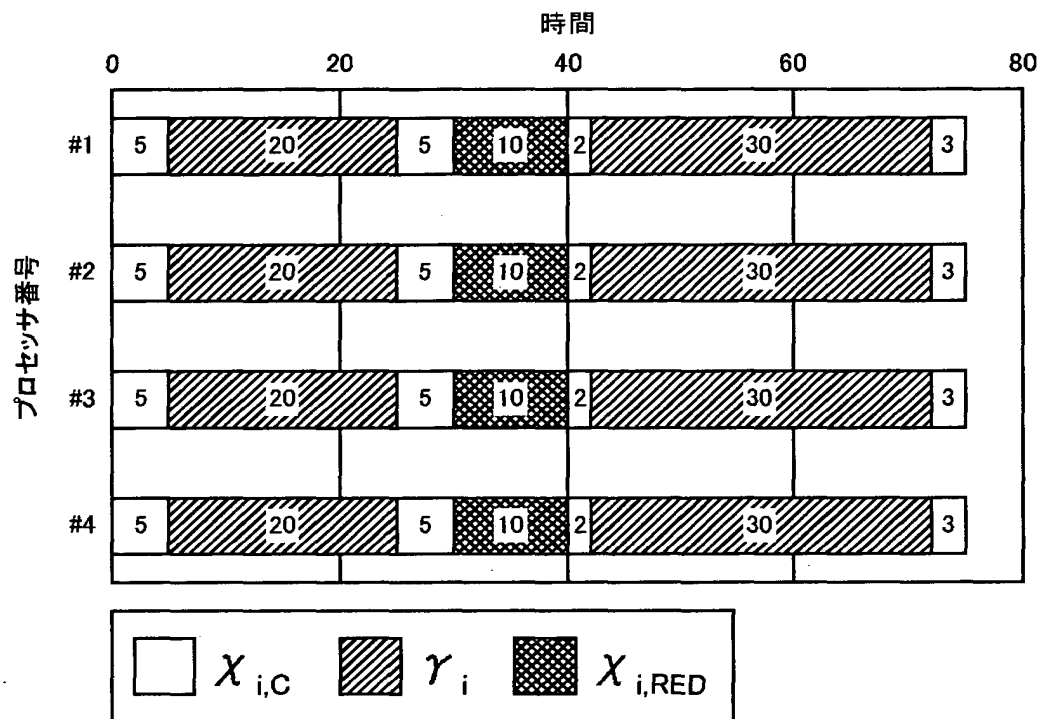
【図 4 6】



【図 4 7】

$R_b(4)$	$R_p(4)$	$A_p(4)$	$R_C(4)$	$R_W(4)$	$E_p(4)$	$E_p(4) \cdot p$
0.7875	1.000	∞	0.1418	0.2778	0.4521	1.808

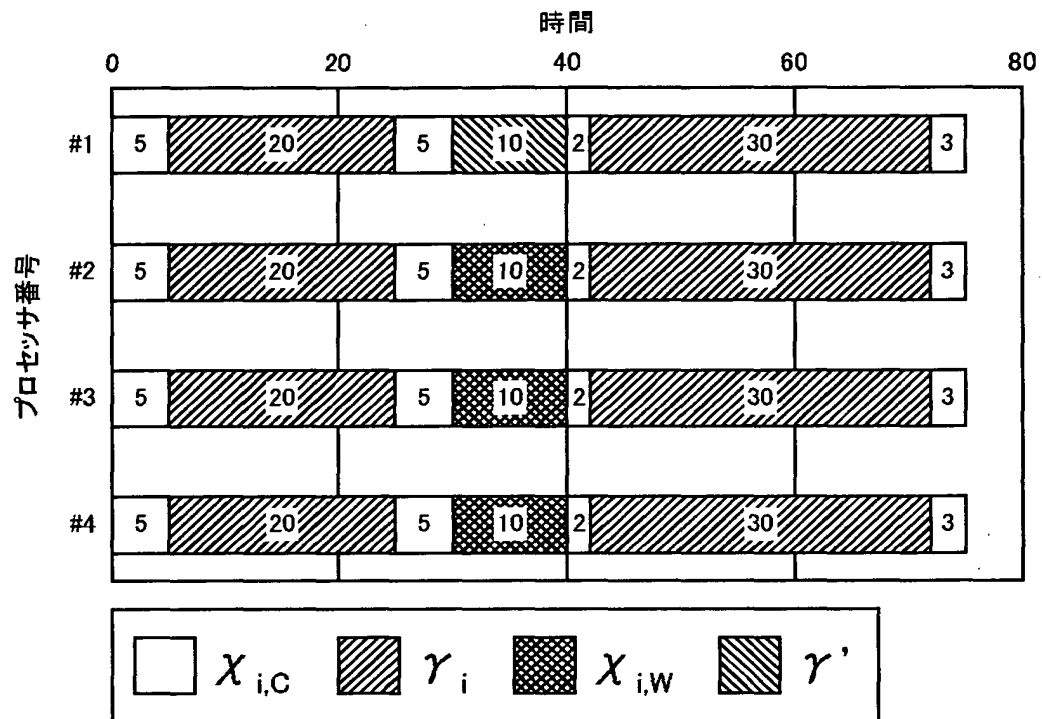
【図 4 8】



【図 4 9】

$R_b(4)$	$R_p(4)$	$A_p(4)$	$R_{RED}(4)$	$R_C(4)$	$E_p(4)$	$E_p(4) \cdot p$
1.000	0.9524	21.01	0.1333	0.2000	0.7000	2.800

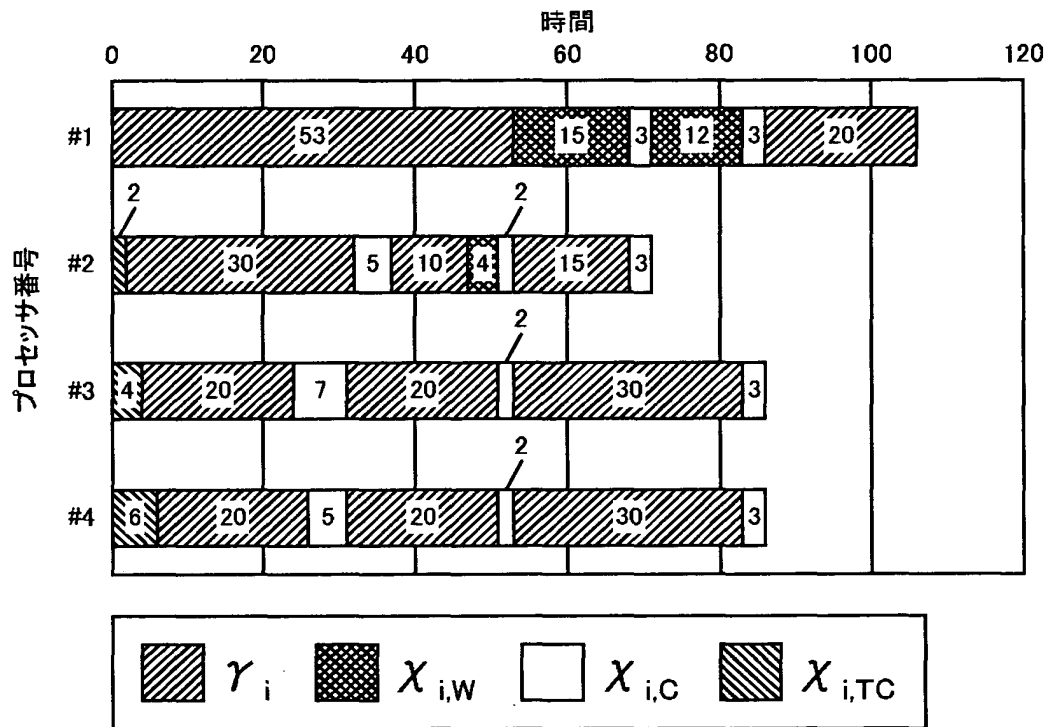
【図 5 0】



【図 5 1】

$R_b(4)$	$R_p(4)$	$A_p(4)$	$R_C(4)$	$R_W(4)$	$E_p(4)$	$E_p(4) \cdot p$
1.000	1.000	∞	0.2000	0.1000	0.7000	2.800

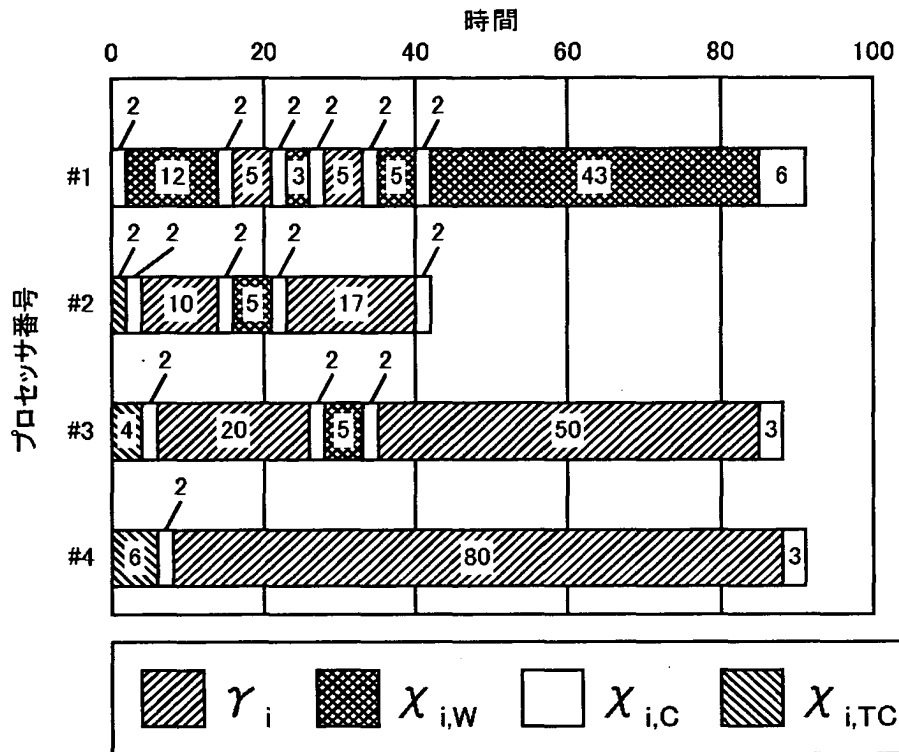
【図 5 2】



【図 5 3】

$R_b(4)$	$R_p(4)$	$A_p(4)$	$R_{TC}(4)$	$R_C(4)$	$R_W(4)$	$E_p(4)$	$E_p(4) \cdot p$
0.8231	1.000	∞	0.0344	0.1089	0.0888	0.6321	2.528

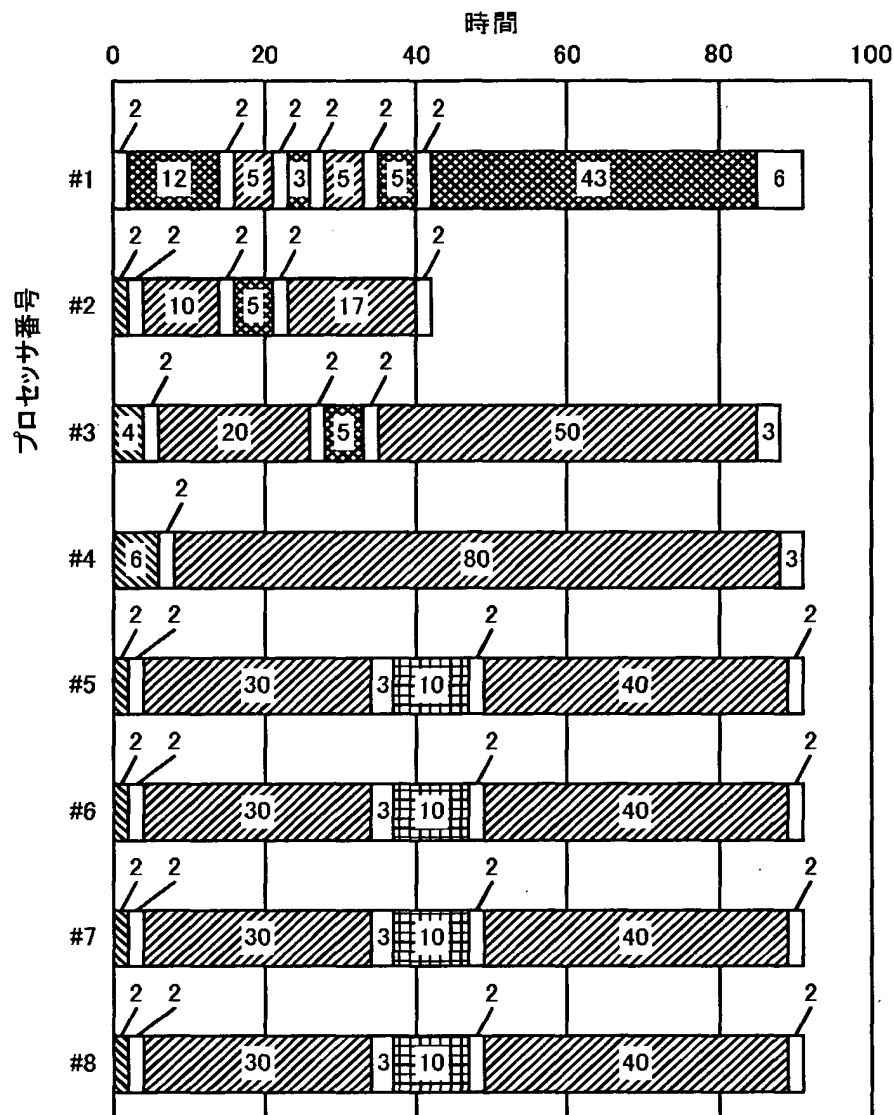
【図 5 4】



【図 5 5】

$R_b(4)$	$R_p(4)$	$A_p(4)$	$R_{TC}(4)$	$R_C(4)$	$R_W(4)$	$E_p(4)$	$E_p(4) \cdot p$
0.8571	1.000	∞	0.0385	0.1282	0.2340	0.5137	2.055

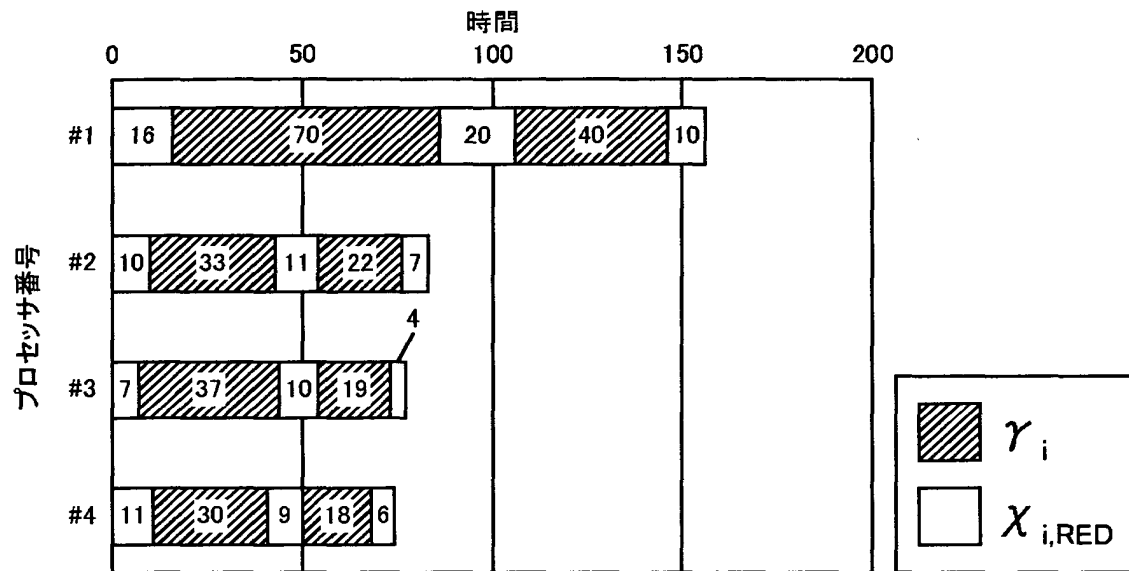
【図 5 6】



【図 5 7】

$R_b(8)$	$R_p(8)$	$A_p(8)$	$R_{RED}(8)$	$R_{TC}(8)$	$R_C(8)$	$R_W(8)$	$E_p(8)$	$E_p(8) \cdot p$
0.9286	0.9790	47.62	0.0592	0.0296	0.1124	0.1080	0.6552	5.242

【図 5 8】



【図 5 9】

$R_b(4)$	$R_p(4)$	$A_p(4)$	$R_{RED}(4)$	$E_p(4)$	$E_p(4) \cdot p$
0.6250	0.8988	9.881	0.3103	0.4796	1.918

【書類名】 要約書

【要約】

【課題】

ロードバランスが保たれていない場合にも、ヘテロなプロセッサ環境を含め多数の並列処理に適用でき、並列効率と並列性能評価指標であるロードバランス寄与率及び仮想並列化率、並びに並列性能阻害要因間の定量的関係付けを行う。

【解決手段】

並列効率 $E_p(p)$ は、並列計算機システムに含まれる各プロセッサ間の負荷の均衡度合いを表すロードバランス寄与率 $R_b(p)$ と、並列計算機システムにおいて実施した処理のうち各プロセッサにより並列計算された部分の、時間についての割合を表す仮想並列化率 $R_p(p)$ と、並列計算機システムに含まれる全プロセッサの全処理時間に対する各並列性能阻害要因部分の処理時間の割合を表す並列性能阻害要因寄与率 $R_j(p)$ とを用いて以下のように計算される。

【数 1】

$$E_p(p) = R_b(p) \cdot \frac{1}{R_p(p)} \cdot \left(1 - \sum_{j=1}^{j_{Others}} R_j(p) \right) \quad (4-4)$$

出 願 人 履 歴 情 報

識別番号 [0 0 0 0 0 5 2 2 3]

1. 変更年月日 1 9 9 6 年 3 月 2 6 日

[変更理由] 住所変更

住 所 神奈川県川崎市中原区上小田中4丁目1番1号
氏 名 富士通株式会社